RADC-TR-69-304
Technical Report
November 1969

ON-LINE RETRIEVAL

Informatics Incorporated

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMIDP), GAFB, N. Y. 13440.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, New York

227

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded, by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

ON-LINE RETRIEVAL

Thomas C. Lowe
David C. Roberts
Informatics Incorporated

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMIDB), GAFB, N. Y. 13440.

FOREWORD

This interim technical report was prepared by Informatics, Inc., 4720 Montgomery Lane, Bethesda, MD, under contract F30602-69-C-0038, Project 4594, Task 459401, for Rome Air Development Center, Griffiss AFB, N.Y. Contractor's report number is TR 69-1090-1. The RADC project engineer was Mr. Murray Burke (EMIDB).

The distribution of this document is limited under the U.S. Mutual Security Acts of 1949.

This technical report has been reviewed and is approved.

Approved: Murray Burke

MURRAY BURKE
Information Transfer Sciences Section
Intelligence Data Handling Branch

Approved: Francis M. Kelly

JOHN A. THOMPSON, Lt Col., USAF
Intelligence Data Handling Branch
Intelligence & Reconnaissance Div.

ABSTRACT

The report is concerned with the development of an on-line information storage and retrieval system for the Rome Air Development Center. It is deeply concerned with automatic document classification, interactive retrieval and the problems peculiar to the automated processing of large document collections. In the light of the requirements to be met, a report on existing applicable techniques is made. This is followed by a description of other methods, developed specifically for this project. Finally, the system design is presented to a detailed level.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd.)

TABLE OF CONTENTS (Cont'd.)

TABLE OF CONTENTS (Cont'd.)

ILLUSTRATIONS

Page

ILLUSTRATIONS (Cont'd)

ILLUSTRATIONS (Cont'd.)

SECTION I

INTRODUCTION

In the field of mechanized information storage and retrieval there are many challenges; ongoing research and development continually advance the state of the art. Three areas in particular are relevant to this report: automatic document classification, interactive retrieval and the problems peculiar to automated processing of large document collections.

This is an interim report on the development of an on-line information storage and retrieval system for the Rome Air Development Center, a system deeply involved with the three areas mentioned above. Of first concern is the recognition of the requirements. In light of the requirements to be met, a report on existing applicable techniques is made. This is followed by a description of other methods, developed specifically for this project. Finally, the system design is presented to a detailed level.

I.1 The Need for Automatic Processing

It is not necessary at this point to enter into a discourse on the "information explosion" that is taking place in government, the military services, and industry. It is sufficient to note that a great need exists for the efficient and effective storage and retrieval of information, and that new methods are required both because of the problem of classifying large numbers of documents and the difficulties encountered in retrieving from large data bases with acceptable precision and recall. A good summary of some problems encountered in manual indexing and thesaurus generation has been made by Dennis, who in part stated:*

*Reference (14), page 68.

1) Cost analysis of any existing manual or computer-aided manual system always showed that the significant part of the budget went to the human editors, indexers, abstractors, thesaurus builders, classifiers, or query formulators required to make the system run. This observation has two implications.

   a) These systems cost too much.

   b) If there really is an information explosion, we are on a self-defeating course. Budgets reflect demand for labor, and the obvious end is that which the Telephone Company saw a few years back when they decided to go to the dial system because otherwise, in the rather near future, everybody in the country would be employed as a telephone operator.

2) Indexing is a monotonous and frustrating job.

3) It is more satisfying to obtain information from the horse's mouth than through an intermediary.

Expanding on Dennis' second point, good manual indexing requires good indexers--and such persons are both scarce and expensive. Indexing is not an occupation with wide appeal. Personnel problems are compounded in certain military situations, where changes in assignments can make the establishment of a competent indexing staff extremely difficult.

The third point does not necessarily reject all manual indexing, but points out that (algorithmic) automatic indexing is necessarily consistent.

So the size of collections presently being encountered, their growth rate, the scarcity of competent indexing personnel and the cost of manual indexing have promoted the search for automated methods of document classification for retrieval (47). This includes not only the actual indexing of documents, but the development of thesauri. Naturally, the development of new retrieval techniques is intimately linked with work in classification methods.

I. 2 Present Technology

It is the purpose of this section to review briefly some of the developments in the field that are relevant to the present effort. The present section is introductory in nature; detailed technical examination is deferred to Chapter III.

A recent survey made by the National Science Foundation (37) shows 118 automated information systems in use, including those simply processing accession lists and performing similar elementary functions. Ninety-eight systems process information retrieval queries, but most of these are batch oriented. Batch information retrieval systems are useful in many applications, but they lack the very important characteristic of rapid turnaround. Tremendous power is gained when the user is allowed to query, inspect results and reformulate his query interactively. This constitutes a simple form of "browsing" in the document collection, and results in the effect called "convergence to high relevance retrieval" by Landau (26).

I. 2. 1 On-Line Systems

Batch system queries are frequently written by a system "expert", who interprets the information requests submitted by system users. But in interactive systems, the user himself formulates queries and operates the system. Therefore, for successful operation, the on-line dialogue must be easy and natural to use.

A user must be able to concentrate on the problems of retrieving information, and not be required to second-guess the designers of the system. Users with differing levels of familiarity are to be expected; the inexperienced user is to be lead through the system step-by-step, while the experienced user should be able to exercise a great deal of flexibility in employment of the system. Indeed, the messages from the user to the system should be terse for the experienced user and more verbose and tutorial for the neophyte.

Much work has been done in the development of on-line systems, and additional background is provided in Section I. 4, where several such systems are briefly described. But all of these systems retrieve by means of simple coordinate indexing, and various embellishments on it. Consider the additional power of an on-line system if the user is given the ability to locate documents that are in some way like a known document. This can be illustrated by considering the problem of finding documents in the stacks of a conventional library. Suppose that one could only request documents by their classification numbers, and that one were not allowed to enter the stacks. Depending on the user's familiarity with the classification system and with the document collection, he might or might not be able to retrieve all the documents relevant to his needs.

Now, if the user can enter the stacks of the library and browse about, his chances of finding useful documents are increased. They are likely to be physically near the documents specified initially (and, of course, may include those documents). The hierarchical classification scheme of the library has been mapped into one-dimensional space: the ordering of the books in the stacks. An on-line system can be built in such a way that the user is free from the constraints of a space of limited dimensionality, and can search for documents "like" a given document. This is known as document-document searching, and is analogous to browsing in a library where every intellectual area (or "concept") corresponds to a different dimension.

Experimental, batch systems using this type of searching have been built, but this capability has not yet been placed on-line. Based on methods proposed by Salton and others, it is now technologically feasible to implement these techniques on-line. Indeed, the system whose design is the subject of this report permits the user to search in this and other manners.

I.2.2 Statistical and Associative Techniques

At this point, it is useful to look at the development of some applicable techniques and to consider document-document searching in slightly more depth. Since much of the present development has its roots in Salton's SMART experiments (40, 41, 42, 43, 44) it is perhaps best to remove any confusion arising from his terminology. The casual reader of the SMART literature might believe that there exists a single SMART system, some fixed entity for information storage and retrieval. Doyle (15) points out that this is not so:

> The word "system" is misleading. It is really a chemistry laboratory for retrieval principles and procedures, in which anything someone might advocate elsewhere can be quantitatively studied, individually or in combination with practically anything else. It is a tour de force in experimentation in the documentation area, the like of which is seldom seen.

Consider now the central problem of information retrieval. Using the term "descriptor" in its broadest sense, the problem is that of retrieving data categorized by one or more descriptors. These query descriptors may be related by logical operators, such as the typical "and", "or", and "but not" found in on-line coordinate retrieval systems. Statistical weighting of the query descriptors is also possible. In either case, it is frequently possible to rank the material retrieved for closeness of match to the query.

Associative and statistical methods may be used to develop a thesaurus of words used to characterize documents. It is interesting, and illustrative of the development of the state of the art, to trace the growth of one such technique. Given the problem of automatically determining descriptors for data consisting of English text, Luhn (33) proposed selecting those words falling in the middle part of the rank-frequency plot. This suggestion, made in 1958 in connection with automatic abstracting, is based on the thought that words of relatively high or low frequency are less useful in terms of resolving power. Seven years later, in connection with legal information retrieval, Dennis (14) obtained some success using a more sophisticated statistical measure. This measure is a function not only of frequency, but of uniformity of distribution. Theoretical considerations on a somewhat more sophisticated algorithm were published by Stone (49) and Stone and Rubinoff (50) in 1968. During this period, Stiles (48) and others were working on association factors relating word pairs.

The generalization from the use of words to word-root forms, or stems, is an obvious one in such work. Less obvious is the use of generalized descriptors, or concepts, rather than words or word-roots only. A concept is simply any attribute of some part or parts of the data base that is useful in discriminating between parts of the data base for the purposes of retrieval. In the present application, documents are such "parts". So a simple thesaurus term is a concept. In order to characterize a collection or data base, a set of concepts may be selected manually or by automatic means. For instance, in the case of a document file, concepts could be the words of "significant resolving power" as determined by one of the automatic means outlined above. (It should be noted that characterization of a large data base containing diverse files is sometimes best accomplished by the use of two or more sets of concepts.)

Every document can be described by a concept vector. The simplest such is the binary vector, where it is indicated that a concept simply is or is not present in the data record. More sophisticated is the weighted concept vector, where a number (generally in the range between zero and one) indicates the degree to which each concept is present. *

* Since the vectors are sparse and of high dimensionality, practical considerations generally dictate that elements equal to zero not be stored.

Of what utility is this approach? Each document is characterized by a vector, whose dimension is equal to the total number of available concepts and whose elements are concept weights or binary concept existence indicators. But any query addressed to the system also possesses a concept vector, and statistical matching of that query concept vector with the document concept vectors produces, for every document, a numeric measure of its relevance to the query. Additionally, given a document of interest, its concept vector may be used like a query concept vector in order to retrieve like documents.

Much experimental work using the concept vector approach has been performed by Salton and others for batch-oriented retrieval applications. But when on-line retrieval is involved, the statistical matching of a query vector against all document vectors may involve an amount of computing which is simply out of the question. Each comparison involves computing some measure, frequently the cosine of the angle between the two vectors. For this and other reasons, clustering of the collection of data into groups (which may or may not be overlapping) may be required. Clustering also aids in a "browsing" type of search.

In order to illustrate clustering, consider a library in which each book is assigned a classification number (Universal Dewey Decimal, Library of Congress, etc.). This number identifies the book uniquely, and contains some information on the contents of the book. Although the number--at a minimum--identifies the position of the book in some two-dimensional, planar tree-structured hierarchy, the books are traditionally placed in the physical stacks in a one-dimensional ordering. If the collection is small enough, one can browse with limited success by examining the neighbors of a known "good" book in the one-dimensional shelf ordering. And one can save time by going directly to the area in which books of interest are likely to be found. But, clearly, a book well described by two different content codes can be in only one physical location.

Conceptually, the location of a document in concept vector space is the tip of its vector. Since the dimensionality of the space is equal to the number of concepts, similar documents become neighbors. The user, then, strolls through a hyperdimensional library in his browsing activities.

But there is another, more practical reason for placing like data records together. Once the general identity of a cluster is established in order to satisfy a query, the number of cosines that have to be computed is simply the population of the cluster, not the entire data base. Frequently this reduction in real-time computing is not just convenient--it is necessary.

I.3 Summary

As the preceding section shows, the development of on-line information retrieval systems and information processing techniques has reached a relatively advanced state*. But most present on-line systems rely on various embellishments of coordinate indexing, range searching, and the like. In terms of actual operating retrieval systems, the more advanced syntactical and statistical schemes have been mechanized in the batch environment, using rather small data bases.

Several topics are of great interest in the context of the On-Line System to be developed for the Rome Air Development Center. The use of concept vectors for document characterization and retrieval is one, and in connection with it is the automatic determination of concepts. Clustering enters into the system in two ways: the clustering of documents for rapid search of a large collection, and the clustering of word roots into concepts. The statistical methods described in this chapter are important in the selection of word roots that are most useful in characterizing documents for retrieval. Morphological analysis must be used in order to determine the word roots, given the words.

* For the interested reader, a brief description of several existing on-line systems' salient characteristics is presented in the next section of this chapter.

Both conventional and document-document searching are desirable, under control of a processor for man-machine dialogue. The dialogue should be of sophisticated design, anticipating both the naive and the sophisticated user. The present state of the art provides a good foundation for this design task.

I.4 Additional Background Information: Existing On-Line Systems

Early systems tended to allow very complicated Boolean combinations of search terms, perform the retrieval and then query the user on the categories of retrieved information to be displayed. If the user wanted to modify his query, he had to return as if he were starting an entire new query. As system designers became more familiar with utilization of interactive communications, a trend away from this became evident. The initial query now tends to be simplier, consisting primarily of a request composed of logical unions and intersections--primarily the former. Then the user is told how many documents were retrieved (sometimes on a basis of "hits" on each of the terms). He may then perform repeated intersections with additional terms. This process is called qualification; it allows the user to start with a retrieval of high recall but low precision, and then increase the precision.

Only one type of ranking in such systems--the identification of relative relevance of retrieved documents--exists. It is obtained from a tally of the number of elements of "or" clauses retrieving each document.

Most such systems rely on manual indexing of documents, and retrieve on descriptors. Some allow the use of an on-line thesaurus; some also allow retrieval by title or author's name. A few allow searching on partial words and word phrases. Thus, most present on-line systems rely on manual indexing (except for title and author information) and perform retrieval based on logical connectives. Except for thesauri, little is done to assist the user with synonym problems.

One approach to reducing or eliminating the need for manual indexing is the automatic generation of inverted files for much of the document text. For a large document collection, this requires a great deal of storage, such as is available in Data Cell-type devices. But unless a manually generated, automatic synonym dictionary is used, recall is degraded. Similarly, one might expect that such a "scattershot" approach would make precision difficult to control.

Typically, the man-machine dialogue in such retrieval systems is available in two or three forms. The inexperienced user needs rather verbose machine messages, providing cues and being somewhat tutorial in nature. As his familiarity with the system grows the reading of such material becomes more of an annoyance than an aid, so a more terse dialogue is required.

By way of continuing this review of present on-line systems, five specific examples are considered briefly below (1, 4, 18, 19, 36, 52).

I.4.1 DIALOG

DIALOG is a proprietary product of the Information Sciences group of the Lockheed Palo Alto Research Laboratory. It operates on 360/30 computers under DOS, and can handle from five to ten CRT terminals. Either CCI or IBM 2260 displays may be used, and each station also has a remote printer. The mass storage device is Data Cell.

Inverted files are maintained on all fields and values for searching, but the data base can be updated only in a batch mode. Only Boolean searching is permitted, although nesting is available indirectly.

DIALOG makes a very interesting use of the special character (upper case figure) keys. These initiate special command functions. For example, in order to initiate a search, a user enters a single term (index term, author's name, etc.) and presses the EXPAND key. The resulting display shows several terms: the one selected and those immediately before and after the selected term in the alphabetical order of the dictionary. With each term is a temporary number, so that future references during the retrieval do not require typing of the entire term. Also with each term is the number of documents to which it is assigned and the number of related thesaurus terms. The user may then request to see bibliographic and indexing data for documents, or continue EXPANDing. If he does the latter, he can view related terms and thus progress through the hierarchy of the thesaurus. Every term he sees is identified by a temporary number, and at any time he may COMBINE two terms to generate a new temporary number with which a new set of documents is associated. The COMBINE operators are union, intersection and set subtraction. The CRT is used for rapid scanning of terms and bibliographic data, and hard copy shows the user's commands, the identifying temporary numbers of the sets they created, and the number of documents in each set. The lists associated with those numbers are not lost during the "browsing" process, so a user may switch from examining the documents he has retrieved back to more modification of the sets being used for retrieval.

I.4.2 Data Central

The Data Corporation, a subsidiary of the Mead Paper Company, offers Data Central for lease, sale or as a service operating on their computers.

Data Central, which is a descendant of Recon Central*, operates on 360/40 computers using DOS and requires Data Cell mass storage. The designers of Data Central rejected IBM's teleprocessing packages (BTAM and QTAM) and developed their own, called TTAM, for Data Central.

Teletypes, IBM consoles such as the 1050, and the CCI-30 CRT displays are used as remote terminals. When CRT terminals are used, a hard copy printer may be utilized also.

As with most systems, both long and short form dialogue is available to the user. But even with the long form several conventions must be memorized by the user. Data Central's greatest advantages come from freedom of searching. Both arithmetic and logical comparisons may be made, and a "universal" character allows for masked searching. Files are inverted on the word level, so partial word phrases may be used to search for the entire phrase. As sentence number and word numbers are determined, it is possible to search for the occurrence of words within a certain distance of each other.

Data Central allows qualification; that is, reduction of the number of hits after an initial query by performing additional intersections. The user can back up in this process if too many hits are deleted. Data Central performs a very limited stem analysis, and the Data Corporation indicates that synonym dictionary lookup could be added. Presently being developed is a limited capability to update files on-line by dribble posting and more sophisticated on-line updating is planned. Extensive diagnostics are provided for data checking during batch file building, and multilevel security classifications are allowed. Output formatting is accomplished by user selection of one of several standard formats, and provisions are made for users to code special formats in COBOL for later selection.

* Not to be confused with NASA's RECON I and RECON II.

I. 4.3 Orbit

A proprietary product of the System Development Coroporation, Orbit has evolved from previous SDS systems: Colex, Circol, and Circ. It operates on 360/50 and 360/65 computers under SDC's ADEPT, and presently uses disk packs for mass storage. Teletypes, IBM consoles such as the 1050 and the CCI-30 CRT display are used as remote terminals, but, unlike Data Central, formats have not been designed for convenient operation of the CCI-30.

The long form man-machine dialogue in Orbit is easy to learn and use, and shorter forms are available for the experienced user.

Orbit's primary files are inverted on the field level, so exact matches are required and partial word phrase searching is impossible. After a preliminary search request, Orbit gives quite complete data on the hits caused by the various terms in the query and allows for extensive qualification. Should a qualification prove too drastic--deleting desirable as well as unwanted documents--the user can revert to the status of retrieval sequence immediately before execution of the last qualification order.

Orbit has no security features, although ADEPT can provide them at the file level. Presently, the only on-line file update possible is the modification of existing records, but more is planned. During batch file building only limited diagnostics are available.

I. 4. 4 TDMS

Another interesting system is the TDMS of the System Development Corporation, which operates under ADEPT on IBM 360/50 and 360/65 computers. Disks are utilized for mass storage, and in addition to a CRT console the available terminal devices include the IBM 2741 teleprinter, Teletype model 33 and model 35. TDMS is descended from LUCID, but includes hierarchical file structuring. Most of the comments here also apply to RFMS, which is to a great degree a reprogramming of TDMS for the CDC 6600 by the Computation Laboratory of the University of Texas.

TDMS is somewhat like an information retrieval system and somewhat like a data management system. Emphasis away from information retrieval in its usual sense is illustrated by the fact that some of the last features planned for inclusion in TDMS include keyword searching and a data format for text. On the other hand, the use of inverted lists, chained in an elaborate hierarchy, and the fact that the system is operated on-line make it look like an information retrieval system. But TDMS can do much more than retrieve--it can perform elaborate processing and manipulation of data.

Access security exists, but only at the file level. Files may be built, modified and deleted on-line, and it is possible to obtain an audit trail. An on-line tutorial is available at any time for the user who gets "lost" in his processing.

I.4.5 CCA 103

The Computer Corporation of America has a system known simply as "103". The CCA 103 operates under DOS on IBM System/360 computers, utilizing IBM 2260 cathode ray tube display consoles. These displays have keyboard input but there is no provision in this system for a local printer. Convenient features of the system include a "hurry mode", in which documents stream through the display at about human reading speed, and a provision to hold the past twelve screen loads in case the user wants to back up. Additionally, temporary files are held so that a user may reuse or modify a previous inquiry. Queries can include only the operators and, or, not, = and ≠, but nesting is permitted. About the only arithmetic operation available is hit count determination. On line file modification is possible. File addressing is accomplished by hash coding on selected file contents.

I. 4. 6 MSISL

The Moore School Information Systems Laboratory* system is, like SMART, an experimental test bed. Unlike SMART, it is used exclusively for development of on-line systems. Although the system utilizes coordinate indexing and inverted lists, it can process queries written in a restricted natural language ("Easy English"). Thus, "Please find for me books concerning statistical functions or standard deviation, but not business oriented entitled 'runcible' 'I'" is an acceptable query. The system developed in this report can also process such queries, but in a totally different manner. The MSISL system performs a syntactical scan of the query and generates an intermediate form: keywords joined by set addition, subtraction and intersection operators, and a special character that permits searching on partial word phrases.

* University of Pennsylvania, The Moore School of Electrical Engineering. See references (75, 76).

SECTION II

THE ON-LINE SYSTEM - OVERVIEW

This report is concerned with the study leading to the design of an information storage and retrieval system, and the design itself. For convenience, the object system is referred to here as the On-Line System. This chapter serves to introduce the reader to the fundamentals of the On-Line System, while following chapters cover the study and design in detail. Thus, the present chapter presents no rationale for the design. Rather, it is intended to provide background information on the final design in order to give the reader some insight into the central question that will no doubt arise during a reading of the subsequent chapters: "What does all this lead to?".

II. 1 Automatic Thesaurus Generation

Although the On-Line System operates on free text, it must use a thesaurus. The thesaurus contains word stems, rather than words, and is automatically developed from the data base. First, common words (a, an, the, ...) are removed and a stem analysis routine employed in order to select the distinct non-common stems occurring in the document collection. This large list of stems is reduced to a smaller collection of so-called content stems, which collection constitutes a thesaurus. The selection of the content stems from the collection of raw stems is performed by the statistical filtering program, which selects those word stems most promising for the characterization of documents. It does so by analysis of both the stem rank-frequency distribution and the variation of that distribution over the document collection.

II. 2 Concept Vectors

With every document is associated its concept vector. This vector consists of concept-weight pairs. A concept vector can be formed from any body of text, so in order to process a retrieval query it is only necessary to derive the concept vector for the query and correlate it with concept vectors for the documents in the collection. Those documents with vectors producing the highest correlation are then retrieved.

The concepts themselves could, of course, be word stems. But this would not allow the system to account for the use of words that are similar in meaning, and would introduce one of the worst drawbacks of simple coordinate indexing: the need for the user of the system to consult a thesaurus of "use" and "used for" terms. Instead of this stem-per-concept approach, the system is to cluster stems into about 1500 groups. Each group contains stems of similar semantic value, and each group corresponds to a concept. The clustering is performed on a basis of statistical stem co-occurrence analysis.

II. 3 Retrieval

An ordinary retrieval on the basis of a text query is performed in the following manner. First, the user's request is processed by the routines which reject common words and perform stem analysis, reducing the query to a sequence of stems. Each content stem is then mapped by a dictionary processor into one or more clusters. Since each cluster is associated with a concept, this process produces the concept vector corresponding to the query. This vector can be correlated against the concept vectors for the document collection in order to perform the retrieval. In order to avoid comparison with all the concept vectors for a collection of potentially 40, 000 documents, the document concept vectors themselves are clustered about centroids. This materially reduces the search time.

As mentioned in the last chapter, document-document correlation can also be performed by the On-Line System. This form of searching simply employs the concept vector of a known document in order to retrieve similar documents. (It is also possible for the user to construct and modify query concept vectors directly, working only with numeric concept codes and weights.)

During the retrieval process, the user can be expected to try a number of queries. Some will retrieve desirable documents, and some will not. The user is given the capability to build a file of documents, retaining those which he finds desirable.

SECTION III

TECHNIQUES FOR AUTOMATIC CLASSIFICATION AND RETRIEVAL

This chapter is concerned with the basic techniques to be employed. Details of mechanization and the determination of design parameters are discussed in following chapters, as are all aspects of interactive retrieval via man-machine dialogue. A brief overview of the system is presented in Chapter I.

III.1 Concordance of Stems

Present estimates place the size of the data base at 40,000 psychological abstracts, each containing no more than 3000 characters of text information. This means that the data base consists of up to 120 million characters. Written on magnetic tape with one inter-record gap per foot of tape, recorded at 556 bits per inch, the data base would require nine 2400-foot tapes.

A data base of this size cannot be manipulated extensively without consuming exorbitant amounts of computer time. For example, simply reading the entire data base at 18,000 characters per second requires nearly two hours. On the 635, sorting an input file that occupies three reels requires 14 tapes as intermediate collation units. Thus, a sort of this much data would require a large number of tape mountings and dismountings by the computer operators, not to mention the use of perhaps ten hours of computer time.

It is easy to see, then, that multiple-pass processes cannot be made starting with the entire data base. Any programs that read the original data base must be sequential-access, one-pass operations. The most desirable approach to the entire preliminary processing task is to produce some intermediate data structure containing only the information needed for the remaining processes, and then use that intermediate structure for all subsequent processing.

One such intermediate form of the data that could be used is a concordance of the text. A concordance is an alphabetical index of all the important words (or, as in the present case, word stems) in a document. Before the computer age, detailed concordances were constructed only for works whose content was to be subjected to intensive analysis, such as the Bible. These manually generated concordances listed only the major occurrences of a small subset of the total vocabulary used in the text. Even then, the generation of a concordance was a tedious and costly undertaking.

Using the computer, a concordance can be prepared with much greater detail and accuracy than could ever be achieved manually. The computerized production of a concordance is usually a simple procedure. A simple program is used to recognize individual word stems in the text and tag each word with its location; each word-tag pair is then written onto tape. The pairs are then sorted, using the word as the primary field and the tag as the secondary field. A simple merge pass completes the process. If the merge program also counts identical occurrences, and includes these counts in the final concordance entries, then a weighted concordance is produced.

Figure III-1 illustrates concordance preparation. The weighted microconcordance of stems, described in subsection III.1.1, is an intermediate form of the concordance that is useful for other data preparation processes.

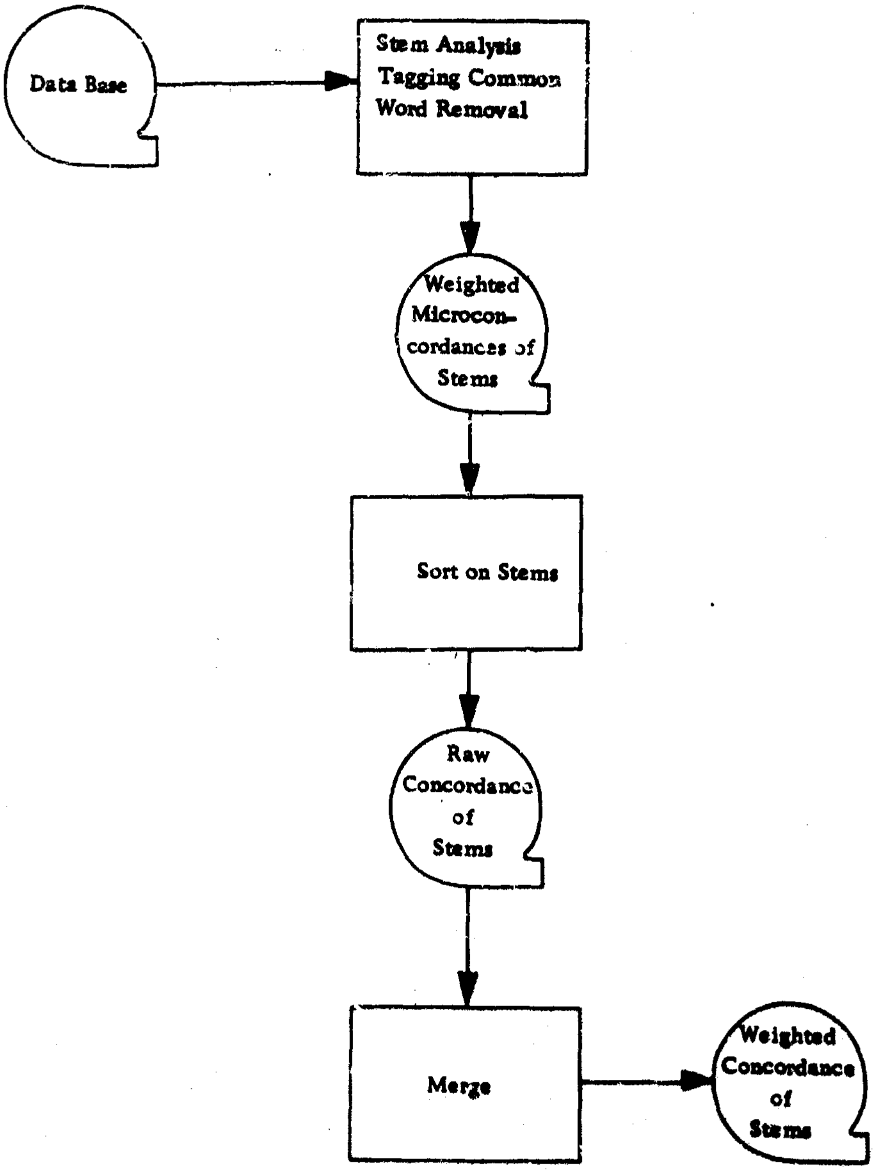

Data Base
Stem Analysis Tagging Common Word Removal
Weighted Microconcordances of Stems
Sort on Stems
Raw Concordance of Stems
Merge
Weighted Concordance of Stems

Figure III-1. Concordance Preparation

A weighted concordance of the data base greatly simplifies the remaining work in preparing the data base for entry into the system. The concordance is useful for identification of words of high information content, recognition of co-occurring words, generation of the concept dictionary, and the generation of concept vectors. Also, the concordance is much smaller than the data base.

The concordance may be represented by a matrix W, with every row corresponding to a distinct document and every column corresponding to a distinct stem. In that matrix w_{ij} is the occurrence weight of the j^{th} stem in the i^{th} document.

Since W is expected to be sparse, it is not practical actually to use a matrix representation. Instead, the concordance is a collection of entities, one for each stem: the stem itself and a document-weight pair for every occurrence of the stem. Letting S_j be the explicit alphanumeric representation of the j^{th} stem and a_i be the i^{th} document's accession number, then a concordance entry for the j^{th} stem is:

$$(S_j; a_{i_1}, w_{i_1j}; a_{i_2}, w_{i_2j}; \ldots; a_{i_r}, w_{i_rj}),$$

where the only nonzero elements of the j^{th} row of W are those elements corresponding to the i_1, i_2, ..., i_r documents.

III.1.1 Microconcordances

Two forms of weighted concordance are required for this application. There is the total concordance, covering all documents, mentioned above. In addition, a concordance is generated for each separate document in order that the documents may be automatically classified, and for the statistical filter (described in Section III. 2.1.1). This set of microconcordances is actually generated first, and then processed in order to produce the total collection's concordance. In connection with the microconcordances, another parameter is needed for programs performing statistical calculations: the length (in stems) of each document. These data are placed in the relevant microconcordances.

Figure III-2 illustrates schematically the concordance and microconcordance scheme; design details are included in a later chapter.

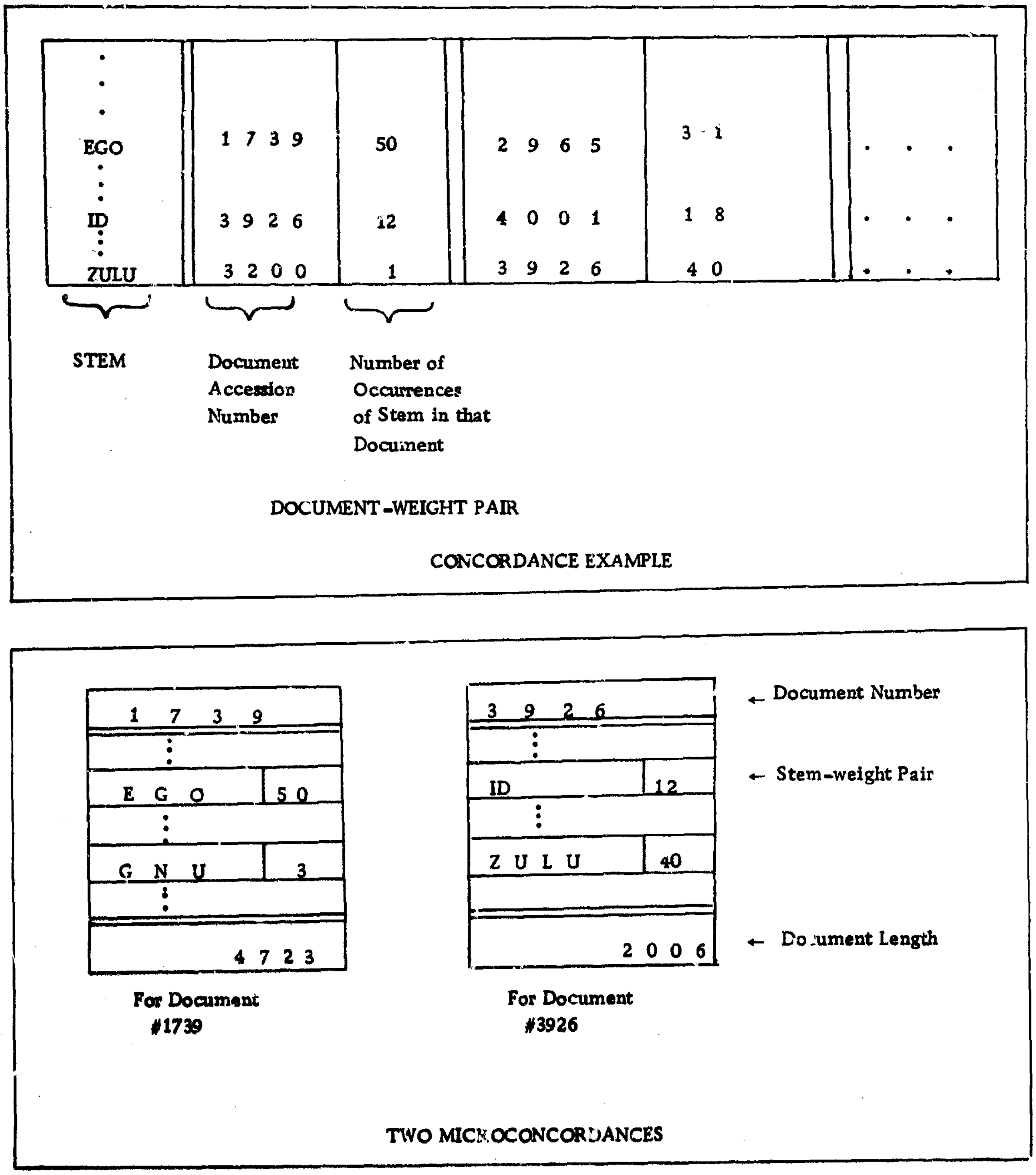
III.1.2 Common Words and Stem Analysis

The concordances mentioned in the preceding section and the dictionary described in following sections are based not on words, but on word stems. The conversion of words to their stems--one might say to their canonical forms--is performed by the stem analysis routine.

Before stem analysis is performed, the computation required and file size can be greatly reduced by rejection of common (or "function" words) such as "the", "a", "at", etc. This section describes the principles of the stem analysis involved; lists of suffixes, common words and other design information are given in Chapter V.

The general objectives lead to two incompatible design goals for a stem analysis program. The first is the need to find stems that are as simple as possible, so that all forms of a given word will be treated as occurrences of the same word; the second is to avoid the generative ambiguity, and consequent association of unrelated material, that result from excessive truncation. For example, "rings" and "ringing" should be stored as "ring", with the suffixes "ing" and "s" removed, but removal of the common suffixes "ed" and "ing" from "wed" and "wing" would cause them both to be associated with the stem "w". Thus, some sort of compromise must be struck.

The literature is full of examples of generative ambiguity caused by suffix removal. All examples, however, use short words, such as the



CONCORDANCE EXAMPLE

TWO MICROCONCORDANCES

Figure III-2. Schematic Illustration - Concordances and Microconcordances

"wed" and "wing" example above. Many of these problems can undoubtedly be removed by prohibiting suffix removal that would produce a stem of fewer than some mininum number of letters. What that number should be is an interesting topic for investigation. Six seems reasonable as an upper bound, since it corresponds to the word size of the 635/645; four might be acceptable. Three is probably too small, and two is definitely too small.

This algorithm combines the techniques used by various investigators; the list of suffixes in Section VI.1.1 is a similar composite.

1) In all the following steps, no action that would produce a stem of fewer than n characters is performed, where n is specified at run time.
2) Compare the terminal characters of the word with the list of suffixes; if a match is found remove the suffix from the word. The comparison begins with the longest suffixes in the list and ends with the shortest.
3) If a suffix was removed in the preceding step and the remaining stem terminates in a double consonant, remove one of the consonants.
4) If the suffix "ly" was removed in step 2) and the word's terminal letter is "i", delete the "i".
5) Repeat steps 2) to 4).

The test of suffixes must begin with the longest suffixes. If short suffixes are deleted first, long suffixes will not be recognized. A partial exception to this is the suffixes " s ", " 's ", and " s' "; these can be affixed to words that already contain other suffixes, so there is some justification for first checking for these. However, initial removal of terminal "s" complicates recognition of suffixes ending in "s". The repetition of step 2) and step 5) permits removal of " s ", " 's ", " s' ", and one other suffix; this should be sufficient.

III.1.3 Maximum Stem Length

The processes of automatically classifying documents and retrieving them requires the application of a dictionary lookup processor to a dictionary of stems. It is convenient for the stems concerned to be contained in fixed length fields, and in this subsection a field length and dictionary lookup method are established.

Lowe (28) has considered retrieval by truncated English words, and developed formulas for the number of file accesses required when artificially generated homographs cause multiple accesses to be required. Empirically testing with a relatively small data base, his results show a very rapid decrease in the required number of accesses, directly related to the generative ambiguity, as the number of characters used approaches six. Dennis (14) has tested a thesaurus of 2400 words in order to justify her use of six characters in an information retrieval system. Truncation to six characters generated artificial homographs in only 1.6% of the words; there existed natural homographs for 16%.

In a scientific vocabulary, words tend to be longer and, therefore, more characters are needed--consider the words beginning with "electro..." and "psycho...". Zunde and Dexter (54) investigated the distribution of differing length words by studying the 5153 most commonly used words in several popular magazines, a 3210 word Interagency Life Sciences vocabulary, and a 3315 word NASA vocabulary. The distribution of lengths for the latter two peak at eight and seven characters, respectively. The words from the popular magazines peak at five characters. The distributions all fall off rapidly. Fewer than five to six percent of the Life Sciences and NASA vocabulary words exceed twelve characters, so the number of artificial homographs generated by truncation to twelve characters must be quite small.

The above results from the literature certainly support the use of a twelve character maximum for most applications. In the present case, two other facts add overwhelming evidence that twelve is an adequate number. First, the results presented above are based on words, not stems.

Stems are shorter than words, adding an extra margin. Secondly, the generation of a few artificial homographs adds ambiguity which in turn decreases precision in this type of system. But the homographs do not cripple as they can in the case of ordinary coordinate indexing that uses only complementation and intersection.

III.2 Concepts and Concept Vectors

In the On-Line Retrieval System, each document will be indexed by the use of concept vectors, similarly to the fundamental text analysis processes of the SMART system. First, concept vectors are derived from texts. A concept can be a stem or a set of stems. A sequence of concepts produces a vector; in its simplest form, this vector can be a list of key words that can be used to identify a document.

Concepts can be generated in many ways. For example, SMART's procedures can be divided into two classes -- syntactic and statistical. A recent paper (43) describes experimental evaluation of SMART's statistical features. Early use of syntactic analysis has been disappointing. The primary reason for this is the excessive processing time required for the Kuno syntax analyzer. Worse yet, the timing is very erratic. Although an improved version of that analyzer is under development, no extensive experiments using syntatical analysis have been reported. Further, Salton has concluded that absolute accuracy of a few items is not as good as many "correctly analysed" items and, therefore, that statistical procedures will probably be superior to syntactical ones (42). Because of this, the decision has been made to include only statistical processing in the On-Line Retrieval system

The generation of a concept vector for each document in the data base includes three distinct processes, as illustrated by the flow-chart of Figure III-3. The elimination of non-content stems is performed by the statistical filter discussed in III.2.1.1. Following this, occurrence correlation is used to identify concept centers, and construct a concept

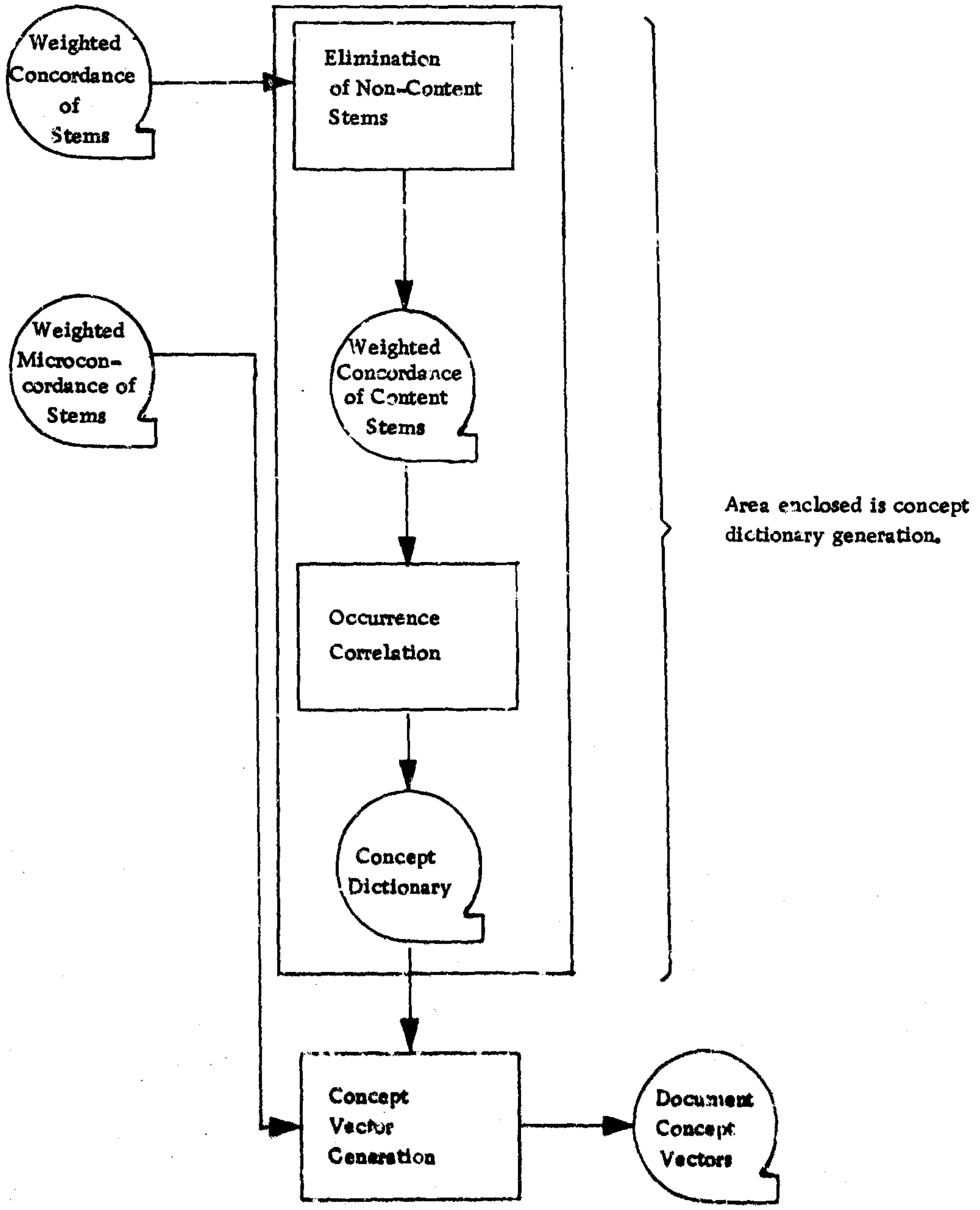

Weighted Concordance of Stems
Elimination of Non-Content Stems
Weighted Microconcordance of Stems
Weighted Concordance of Content Stems
Area enclosed is concept dictionary generation.
Occurrence Correlation
Concept Dictionary
Concept Vector Generation
Document Concept Vectors

Figure III-3. Concept Vector Generation

dictionary, as described in III. 2. 2. 3. Finally, the dictionary is used to generate a concept vector for each document, using its microconcordance, as described in III. 2. 2.

III. 2. 1 Concept Identification

Concepts will be identified by the use of two distinct processes; statistical filtering and occurrence correlation. The statistical filter will remove stems that are relatively uniformly distributed throughout the document collection in relation to their absolute frequency, and produce a weighted concordance of content stems. Concept clustering will select as concept centers those stems whose occurrence vectors correlate highly with the occurrence vectors of several other stems.

The stem analysis and common word routine will remove only very common words that obviously convey no information. Other words, however, may occur in such a large subset of the documents in the collection that they are of little value in differentiating between documents. Since not all stems in the collection are of value in distinguishing between documents, certain stems will be ignored. These words have been called "noncontent" words because of their apparent lack of association with the varying specialized content of different documents.

The simplest technique for this purpose would be to remove the words of highest total number of occurrences in the collection, because these would probably also be the most widely distributed. More sophistication could be obtained by using a bandpass technique on the rank-frequency distribution, and discarding the most and least common words.

Some measure of the uniformity of the distribution of words throughout the collection, in addition to the number of total occurrences, is intuitively attractive. A uniformly distributed word with many occurrences would not be useful in distinguishing documents, and this word should be discarded. On the other hand, a uniformly distributed word with a low total number of occurrences would appear in only a small number of documents, and should be retained. Two very similar approaches to this problem have been suggested by Dennis (14) and by Stone (49) and Stone and Rubinoff (50).

The occurrence correlation program will read the weighted concordance of content stems, and write a tape containing the concept dictionary. This dictionary will list, along with each stem in the concordance of content stems, the concept numbers into which it is to be mapped, and the weight associated with each mapping.

Words that often co-occur will be mapped into common concepts during concept vector generation. The occurrence correlation program will compute a correlation coefficient between each pair of content stems in the concordance; the pairs of greatest correlation will be assigned identical concept numbers. Because each word's occurrences must be correlated with those of each other word, the entire correlation process must be performed in core if it is to use a reasonable amount of computer time. Thus, the size of the vocabulary upon which occurrence correlation can be performed is limited by practical considerations on computation.

Occurrence correlation consists of dividing the set of occurrence vectors into overlapping subsets of similar size. The vectors within each subset should be highly correlated with one another. In this way, the concepts can be used to characterize each document in the collection as a concept vector, where each coordinate of the multi-dimensional concept vector is the weight of a concept occurrence. In order to achieve the simplest possible characterization of documents by concept vectors, it is desirable that the concepts be relatively independent, in the same way that a base for any given vector space provides the most compact representation of that space.

III. 2.1.1 Content Stems and the Statistical Filter. An automatic method of content stem determination forms the basis for the automatic generation of the classification vocabulary. Two approaches are considered: those based on bandpass considerations and on frequency-variance analysis. Although the latter are selected for the On-Line System, bandpass techniques are described and observations on the critical parameters are made. The frequency-variance methods are an extension of the bandpass approach.

III. 2.1.1.1 Bandpass Techniques. For any concordance one can plot the rank-frequency distribution of words (or in the present case, stems). Items of very high frequency can be supposed to contribute little information of significance, while words occurring very occasionally also are insignificant. This is roughly Luhn's reasoning (33), as reported by Meadow (35) and illustrated in Figure III-4. There is no defined measure of the quantity indicated as "significance", and the curve for it indicates only that significance is related to frequency in some way.

For the present system, a somewhat more sophisticated filtering technique than simple bandpass is recommended. But the bandpass approach is appealing and is considered an alternative approach. It is worthwhile to determine what the bandpass parameters involved are, and how they can be adjusted. Cherry (10) quotes a generalised form of Zipf's law (53)

$$\ln p(j) = A - B \ln j,$$

where p(j) is the probability that a word selected at random from the body of text is the word of rank j, A is a constant and B is a constant approxi-

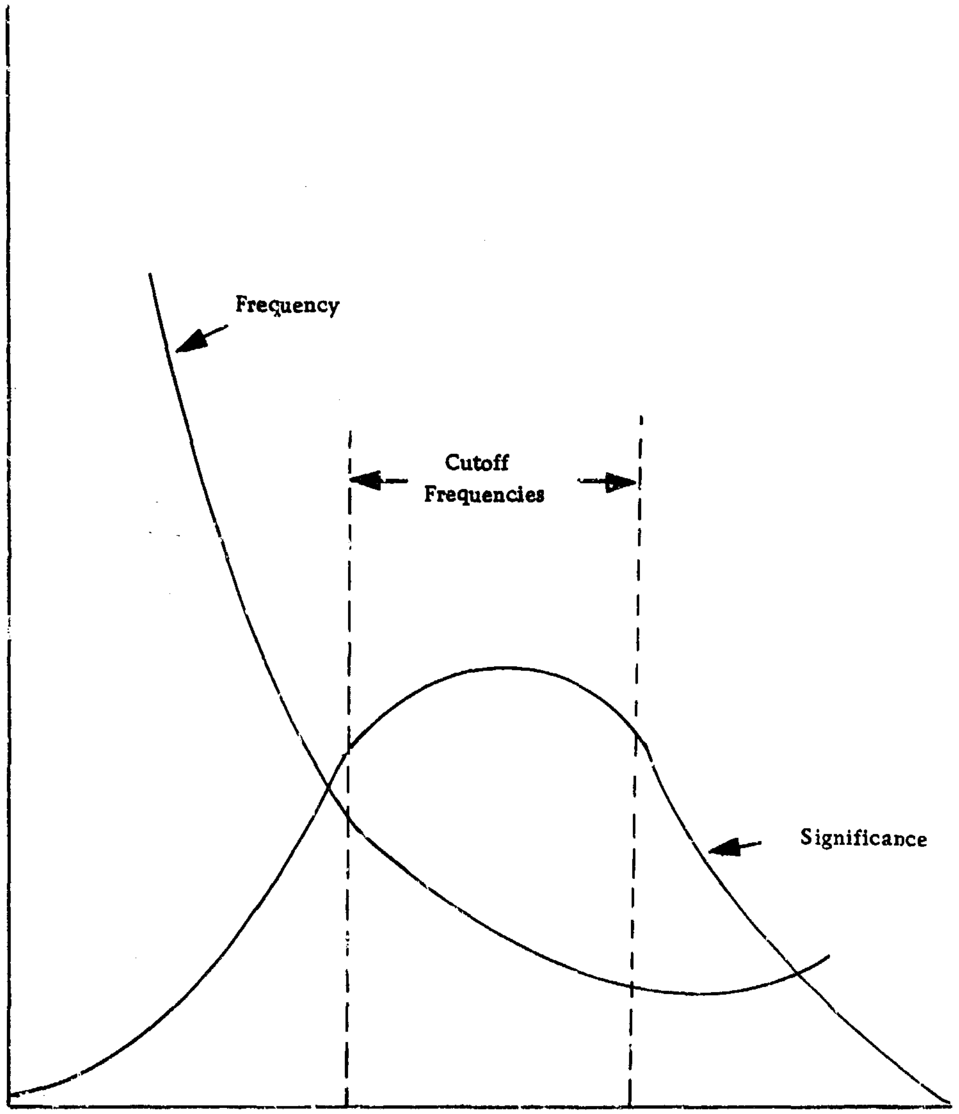


Figure III-4. Rank-Frequency Distribution

mately equal to one*. An equivalent form is

$$p(j) = e^{A} j^{-B}, \text{ or}$$

$$p(j) = K\, j^{-B}.$$

Zipf originally stated his "law" with B = 1 and K = 0.1. Shannon (46) noted that, if the number of distinct words in any examined collection is represented by N, then for <u>any</u> collection Zipf's law requires that there be exactly N = 8727 distinct words. This follows the requirement that

$$\sum_{j=1}^{N} p(j) = 1, \text{ since the only N satisfying the requirement}$$

with $p(j) = 0.1/j$ is N = 8727. Shannon found this assumption adequate for his purposes, as he could safely assume $p(j) = 0$ for $j > 8727$.

In a different application, assuming that N is known and the exponent of j is sufficiently close to unity, Lowe (30) has used the form $p(j) = K/j$. Since N is known and the sum of p(j) over j = 1, ..., N must be one, K can be found and is approximately equal to $(\ln N + \gamma)^{-1}$. (γ is Euler's constant, equal to about 0.5772.)

Thus,

$$p(j) = \frac{1}{j(\ln n + \gamma)},$$

a form useful in analyzing bandpass parameters.

[*] The notation used here is different from Cherry's, for consistency in the following analysis. This generalization of Zipf's law is due to Mandelbrot. Variation of the constants has been noted for texts from different languages, sources, the speech of children and psychotics; see Brillouin (7).

Let the ranks of the highest- and lowest-ranking words defining the bandpass be J_{max} and J_{min}, as shown in Figure III-5. Those words with rank j such that $J_{min} < j \leq J_{max}$ are passed by the filter and therefore are "content" words.

What criteria can be used in the selection of J_{min} and J_{max}? First, the number of content words is $N' = J_{max} - J_{min}$, and the probability that any distinct word is a content word is

$$F = N'/N.$$

A second equation to fix J_{max} and J_{min} can be determined from the probability that a word selected from the collection of text is a content word. This probability is given by

$$F = \sum_{J_{min}}^{J_{max}} p(j).$$

In order to express F in terms of the cutoff ranks, note that

$$F = \sum_{j=1}^{J_{max}} p(j) - \sum_{j=1}^{J_{min}} p(j),$$

$$F = \frac{1}{\ln N + \gamma} \left[\sum_{j=1}^{J_{max}} \frac{1}{j} - \sum_{j=1}^{J_{min}} \frac{1}{j} \right].$$

For $J_{min} > 30$ a reasonable approximation is

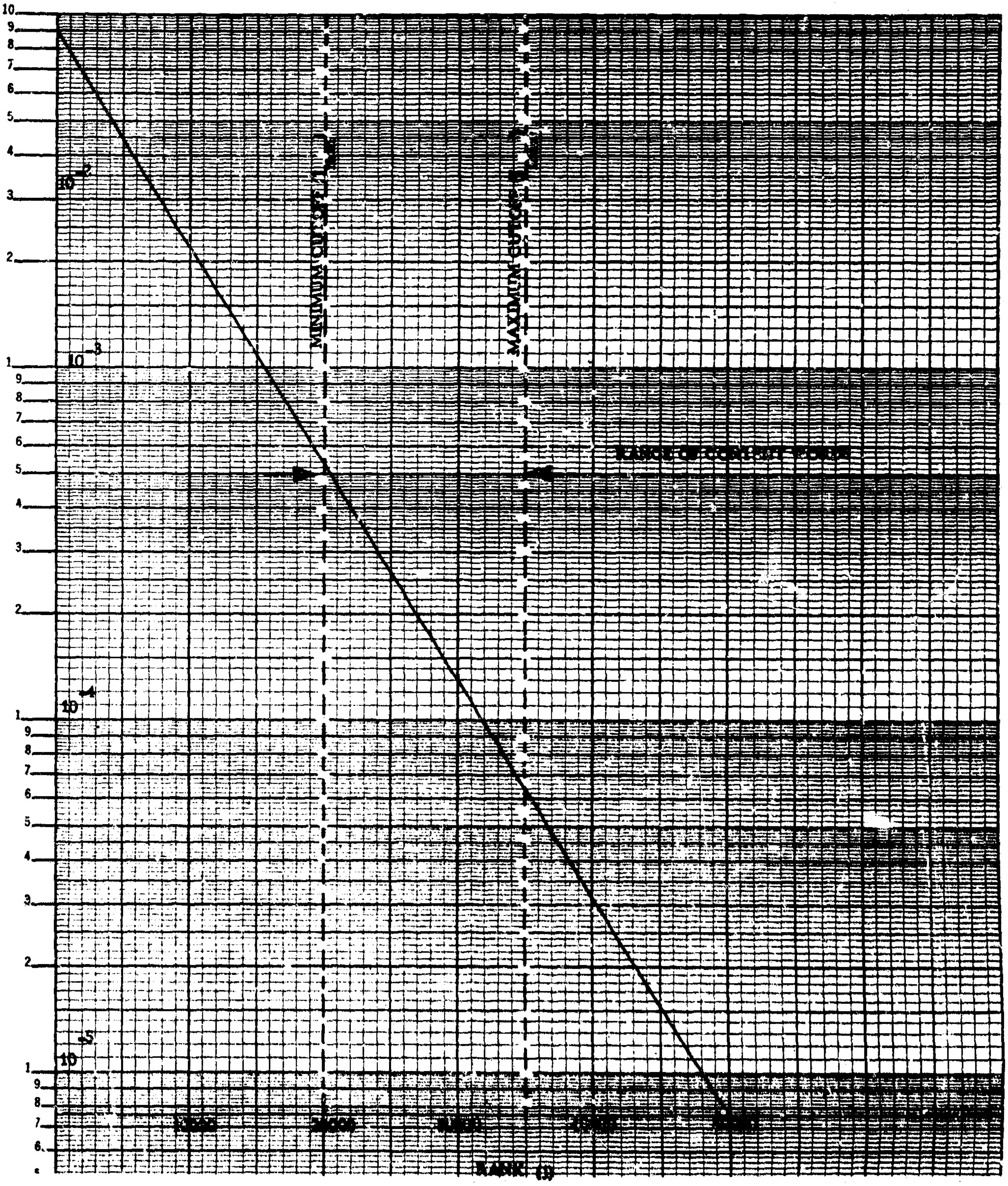


Figure III-5. Setting Cutoff Parameters

$$F = \frac{1}{\ln N + \gamma} \left[(\ln J_{max} + \gamma) - (\ln J_{min} + \gamma) \right],$$

$$F = \frac{\ln \frac{J_{max}}{J_{min}}}{\ln (N + \gamma)}$$

III. 2.1.1.2 Frequency-Variance Techniques.

> "The hypothesis offered here is that word significance is indeed a function of frequency of occurrence, but also of the extent to which this frequency is predictable. If a reader knows that certain words will occur with high frequency then these words are not necessarily the most significant to him. The most significant are the highest frequency words that deviate from the predicted frequency. *Words are significant as subject descriptors, then, in proportion to the difference between their actual and expected frequencies.*"*

That hypothesis has been applied by Dennis (14), and her findings are supported by further work by Stone (49) and Stone and Rubinoff (50). It is to be used in the selection of content stems for the On-Line Retrieval system.

The basic idea is simply that a word appearing with moderate or even high average frequency, but with an uneven distribution over the document collection, is a good candidate for use as a "significant" word. The important feature of this approach is that it takes into consideration the usage of words in *documents,* not simply the entire document collection viewed as a continuous text stream.

* Meadow(35), page 100. Italics are his.

A word used moderately frequently with a rather uniform distribution over the document collection does not help in determination of the salient characteristics of any document. But a nonuniform distribution indicates that the word is useful in discrimination between documents, while a moderately high overall frequency indicates that the word is in general use in the subject discipline.

Dennis determines a single function with a value determined by the number of occurrences of the word in the entire collection multiplied by the normalized variation of within-document frequency. This is generally referred to (for the c^{th} word) as NOCC/EK_c. The details of the computation NOCC/EK_c are given in Section VI. 2. 1.

III. 2. 1. 2 Dimension Reduction. Although it is not shown in Figure III. 3 because it is not necessary for an understanding of the concept identification and concept vector generation processes, there is a minor processing step that is performed between statistical filtering and concept clustering: dimension reduction. Because the occurrence correlation process must be performed subject to practical computation limits, the maximum number of components in an occurrence vector must be reduced*. The amount of reduction that must be performed will be a function of the amount of core storage available and the average dimensionality of the occurrence vectors that remain after processing by the statistical filter.

The observation presented below will be used as the basis for the reduction process. It is noted that if some components of a vector must be discarded, then removing the lowest-magnitude components will introduce the least error into a computation of the cosine between any two vectors.

* Dimension reduction is not the only solution, however. An alternate scheme is considered in Section III. 2. 1. 4.

The problem of selecting the subset of the basis for a vector space that minimizes the average error in forming inner products between any two members of the vector space is not identical to this problem, and is not considered here. A preliminary investigation of that problem indicates that the best approximation to all vectors would require an examination of all the vectors to be approximated. This would require a separate program for this purpose, which is not desirable.

It is desired to find in some m-space an approximation vector $\bar{A}'$ to a vector $\bar{A}$ in n-space, where $m < n$, and where m-space is a subspace of n-space. Further, it is desired that $\bar{A}'$ should be the best possible m-space approximation to $\bar{A}$. "Best approximation" is taken to mean "the $\bar{A}'$ that maximizes the quotient Q, where

$$Q = \frac{\bar{A} \cdot \bar{A}'}{\bar{A} \cdot \bar{A}} \quad ".$$

This observation shows that selecting the m greatest-magnitude components of $\bar{A}$ will yield an $\bar{A}'$ that meets this requirement.

Observation: The best approximation in m-space to a vector $\bar{A}$ in n-space, where $m < n$, is the vector $\bar{A}'$, where the components of $\bar{A}'$ are the m highest-magnitude components of $\bar{A}$.

Proof:

by definition of vector space,

$$\bar{A} = (A_1, \ldots, A_n)$$

Let the components of $\bar{A}$ be renumbered so that A_1 is the highest magnitude component, A_2 is the next-highest, and so on.

Then,

$$\overline{A}' = (A_1', \ldots, A'_m)$$

$$= (A_1, \ldots, A_m) \quad \text{by definition of } A'.$$

To measure the goodness of the approximation, the inner product between $\overline{A}$ and its approximation $\overline{A}'$ is formed:

$$\overline{A} \cdot \overline{A}' = A_1 A_1' + A_2 A_2' + \ldots + A_m A'_m = \sum_{i=1}^{m} A_i A_i'$$

$$= A_1^2 + A_2^2 + \ldots + A_m^2 = \sum_{i=1}^{m} A_1^2$$

This product will be maximized when each of the A_i has the greatest magnitude. But this is how the A_i were selected.

Therefore, $\overline{A}'$ is the best m-space approximation to $\overline{A}$.

III. 2.1.3 Occurrence Correlation. Although occurrence correlation is in fact a clustering operation in which the well-known cosine correlation measure is used, it is unlike the typical clustering problem. This is so because there are estimated to be about 1500 clusters for only 5000 or so content stems, so that the average cluster contains only about 3.5 stems.

Figure III-6 shows the flow of information and the processes involved in dictionary generation. Tape α is the weighted concordance of content stems, and contains for each stem a variable number of pairs (document number, weight in document), and the stem itself.

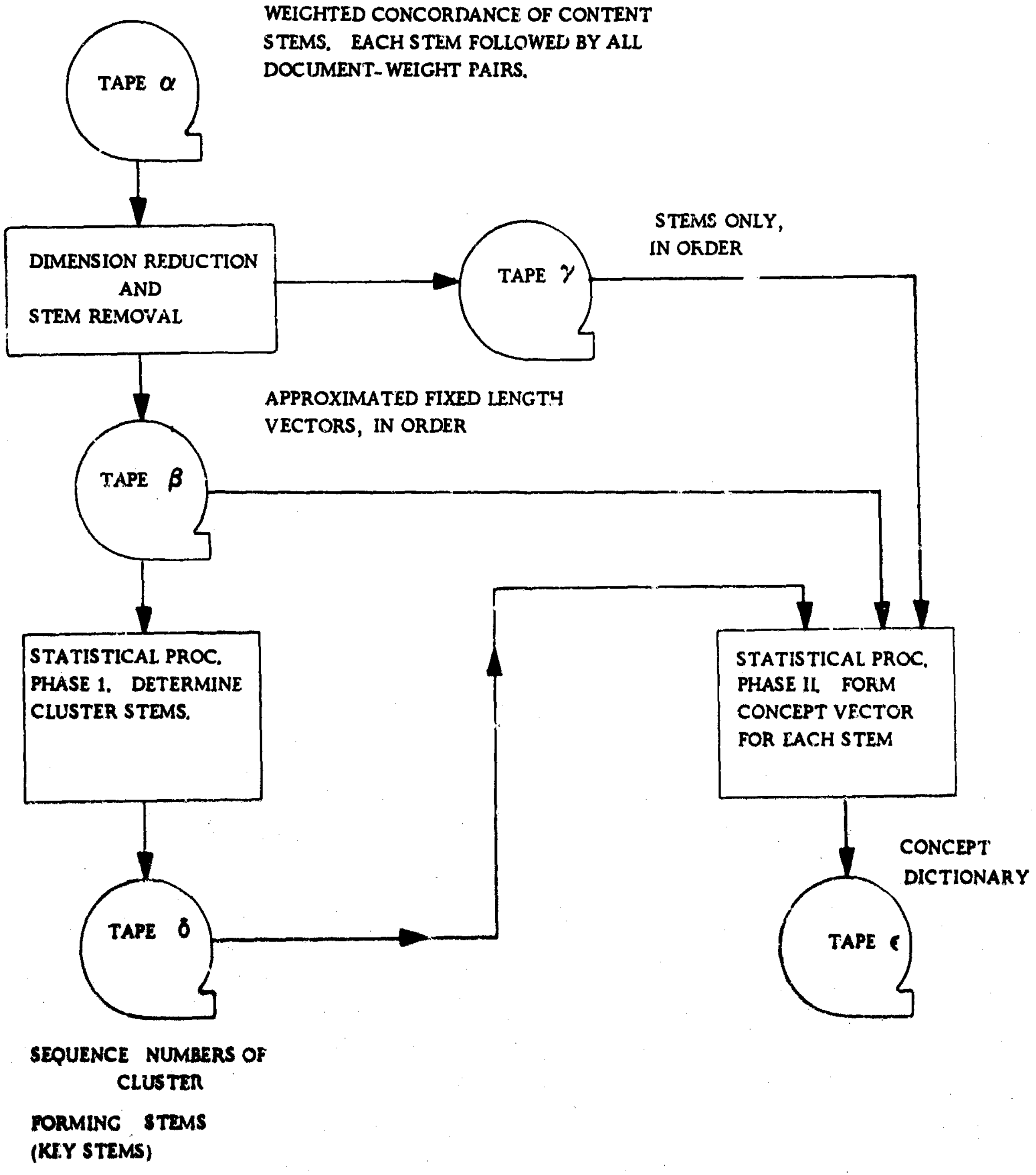

TAPE α
WEIGHTED CONCORDANCE OF CONTENT STEMS. EACH STEM FOLLOWED BY ALL DOCUMENT-WEIGHT PAIRS.
DIMENSION REDUCTION AND STEM REMOVAL
TAPE γ
STEMS ONLY, IN ORDER
APPROXIMATED FIXED LENGTH VECTORS, IN ORDER
TAPE β
STATISTICAL PROC. PHASE 1. DETERMINE CLUSTER STEMS.
STATISTICAL PROC. PHASE II. FORM CONCEPT VECTOR FOR EACH STEM
TAPE δ
CONCEPT DICTIONARY
TAPE ε
SEQUENCE NUMBERS OF CLUSTER
FORMING STEMS
(KEY STEMS)

Figure III-6. Overview of Concept Dictionary Generation

Tape α is passed through a processor (Dimension Reduction and Stem Removal) that is required in order to make in-core processing of the data practical. This program splits stems and their vectors onto two tapes. Since there is a fixed number of stems and they are in lexicographical order, there is no need to retain the alphanumeric stems themselves in part of the processing. Tape γ contains the stems in fixed length records (twelve characters).

The vectors are recorded on tape β, after application of the vector reduction algorithm described in III. 2.1.2. This produces a fixed length vector for each stem. About 25 or 30 document-weight pairs are expected to be used, based on estimates for the size of various programs that must manipulate these data in core and about 70 to 80 thousand core locations available for large batch processing jobs.

On both tapes β and γ, the information is in fixed length records, the ordering being the same as the lexicographical order of the stems. From the 5000 vectors on tape β some 1500 key vectors must be identified, and their sequence numbers (e.g., 27th, 125th, 4867th) written on tape δ. This is done by the process entitled Statistical Processing Phase I.

The approximated vectors for all 5000 content stems, the stems themselves and the identities of the key stems are all fed into Statistical Processing Phase II. It correlates each of the 5000 stems with the 1500 key stems to produce for each stem a concept vector. The vectors are of fixed length, containing perhaps three to five concept number-weight pairs. The concept numbers are initially generated by this routine, which associates concept number one with the first key stem, and so on.

Final output is the dictionary, on tape ϵ.

The processing problems involved here are somewhat different from those of normal clustering, since the average cluster population is only 3.5. In addition, it is desired that this processing be done in core. The method to be used does require calculation of 12.5×10^6 correlation coefficients (cosines), but it does not require their storage at one time.

Suppose that the correlation between two stems is quite high, relative to the set of all cosines generated. Then it appears reasonable to group these two stems together.

In Phase I, first all unordered pairs of stems are correlated. For any comparison a triplet (i, j, c_{ij}) may be formed: i is the sequence number of the first stem, j the sequence number of the second stem and c_{ij} is the cosine of the angle between the two vectors. Since $c_{ij} = c_{ji}$, computations are made only for $i < j$.

Triplets are entered into a table of length LA as they are generated. When the table becomes full, a newly generated triplet (i, j, c_{ij}) is entered in the table replacing an existing (i', j', $c_{i'j'}$) only when $c_{i'j'}$ is the smallest cosine (see III.3, Correlation Measures) in the table ($c_{ij} > c_{i'j'}$). Therefore, at the end of this process the table contains the trip' ts with the largest cosines --LA in number. The table is made as large as possible in order to save computing time for additional passes, but the programs are designed so that if the LA largest correlations are not enough a second pass can be made. This fills the table with the cosine ranking LA + 1 through 2 LA. Depending on available core, the value of LA will probably be about 3500.

Once the table is generated, the triplets can be sorted on their third field. Then the greatest correlation coefficient is first in the table, and so on. From this table, for every tight collection of stems, one representative stem can be chosen as a key stem or concept.

Thus, 1500 key stems are identified. In Phase II every content stem is correlated with the key stems, in order to obtain a short, fixed length dictionary vector. The highest cosines found and the corresponding concept numbers are the contents of this vector, which is the dictionary entry.

III. 2. 1. 4 Alternate Approach Using Binary Vectors. The concept identification problem is covered in the preceding sections, under the assumption that cluster-forming stems are identified by observing the correlation of content stem vectors containing document-weight data of reduced dimensionality. This section is concerned with an alternate approach to the identification of cluster-forming stems; after the identification of the stems the dictionary generation is unchanged. Binary vectors share an advantage with weighted vectors of reduced dimensionality; they require less computer storage than full weighted vectors.

III. 2. 1. 4. 1. Characteristics of Binary Correlation Measures. Salton* identifies several measures in which binary vectors may be used. Conceptually, computing these measures is quite simple. The problems treated in this subsection are those resulting from the mass of data involved.

It is noted that the nature of these measures admits to the partitioning of their calculation. For they all involve terms such as

$$\sum_{i=1}^{D} v_i, \quad \sum_{i=1}^{D} (v_i)^2, \quad \sum_{i=1}^{D} v_i v_i',$$

* Reference (5), p. 236. See also Section III. 3. 2.

which can be computed in sections using identities such as

$$\sum_{i=1}^{D} v_i = \sum_{i=1}^{d_1} v_i + \sum_{i=d_1+1}^{d_2} v_i + \ldots + \sum_{i=d_m+1}^{D} v_i$$

where $1 < d_1 < d_2 < \ldots < d_m < D$.

III. 2. 1. 4. 2 Representation of Binary Content Stem Occurrence Vectors

For each content stem, the binary vector can be represented either by listing the accession numbers* of documents in which the stem appears or by an actual binary vector in which each bit position corresponds to a single document. In the latter case, and assuming a sample of 5000 documents, the vector for each content stem consists of 5000 bits or 139 computer words of 36 bits each.

In the former method, since 13 bits uniquely identify a document within a collection of 5000, two nonzero elements of a vector may be identified in each computer word. Naturally, the number of words required for each vector is variable. For 5000 documents and an average occurrence rate of 2%, the average vector requires 50 words; this requirement is 250 words if the rate is 10%.

Thus, the binary vectors for 5000 content stems in a collection of 5000 documents requires 695,000 computer words if actual binary representation is used, and something between 250,000 and 1,250,000 words if the documents are explicitly identified.

[*] True accession numbers are not actually required. and may actually waste space. Any means of uniquely identifying the individual documents within a chosen sample collection will suffice.

III. 2. 1. 4. 3 Multi-Pass Processing. If the summations for computation of the correlation coefficients are partitioned, the contributions from the first d_1 documents, the next $d_2 - d_1$, etc. are computed. Each of the m + 1 passes computes the contribution of a set of documents to each correlation coefficient, but unfortunately, there are 12.5×10^6 such coefficients. A ranking table technique cannot be employed, since the magnitude of the contribution of each pass to an individual correlation coefficient cannot be determined. Thus, about five tapes are generated as intermediate output from each pass. On the final pass, ranking techniques can be used to determine the most highly correlated content stems, within any desired cutoff level.

The number of passes needed is a linear function of the number of documents involved; the volume of data generated by each intermediate pass varies as the square of the number of content stems.

Again assuming 5000 documents, 5000 content stems and 100,000 words of core available for vector storage, if a binary representation of the vectors were employed the number of passes required would be seven. Figure III-7 illustrates this process.

First, binary concordance data are derived from the weighted concordance. An initial pass through the correlation program produces five tapes containing correlation coefficients based on the first 720 documents (S_1 through S_5)*. On additional passes, the contributions from the other 5280 documents are added in, and one additional scratch tape is required. Finally, the final calculation and ranking are performed.

* The final pass requires dividing by some quantity such as the product of sums of squares in the case of all algorithms except the vector dot products. These data are also totalled, for later division.

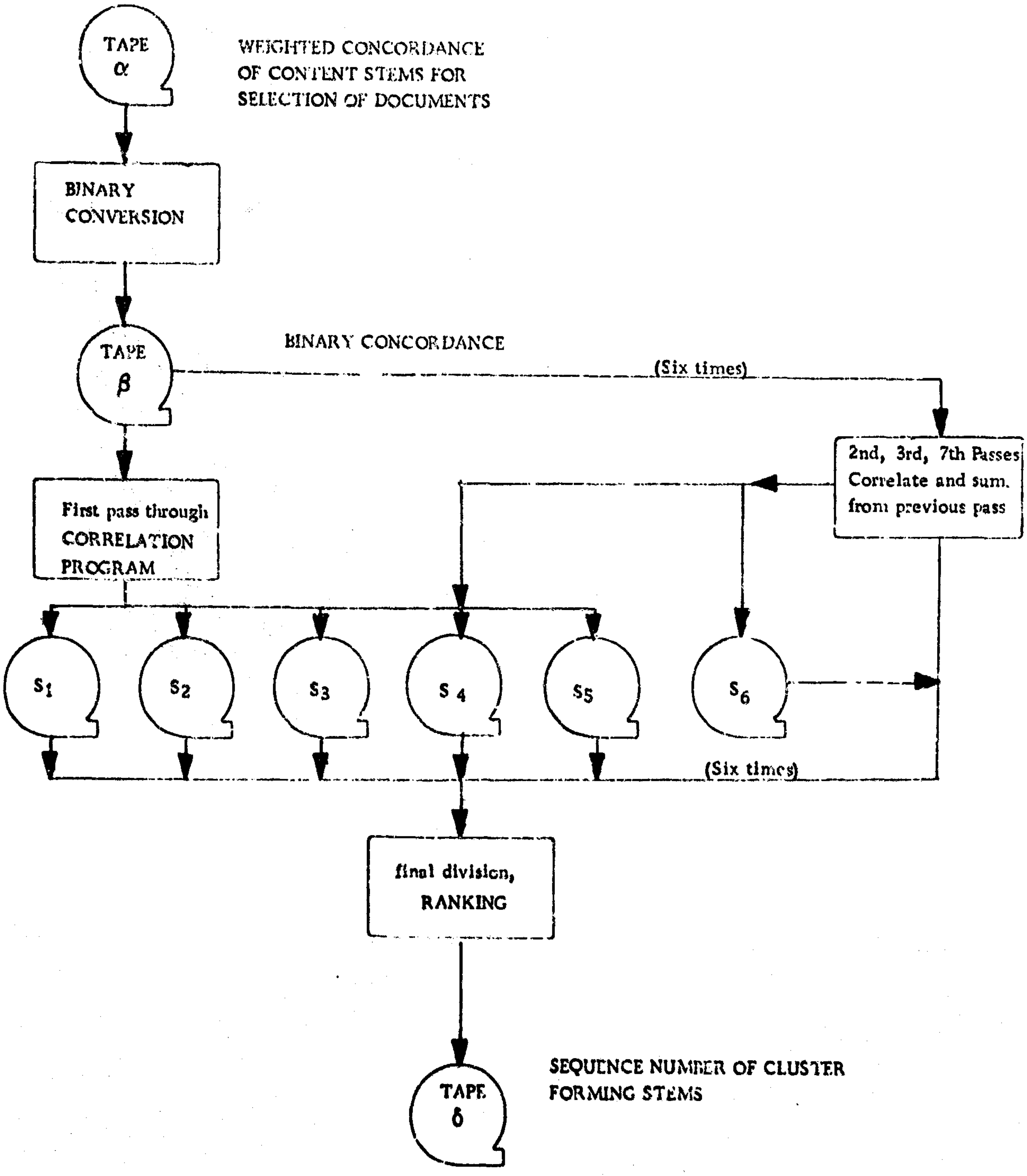

TAPE α
WEIGHTED CONCORDANCE OF CONTENT STEMS FOR SELECTION OF DOCUMENTS
BINARY CONVERSION
TAPE β
BINARY CONCORDANCE
(Six times)
2nd, 3rd, 7th Passes Correlate and sum. from previous pass
First pass through CORRELATION PROGRAM
S_1
S_2
S_3
S_4
S_5
S_6
(Six times)
final division, RANKING
TAPE δ
SEQUENCE NUMBER OF CLUSTER FORMING STEMS

Figure III-7. Content Stem Clustering-- Binary Vectors

A rough calculation indicates that if disk were used for intermediate storage, about 112 million characters would be needed.

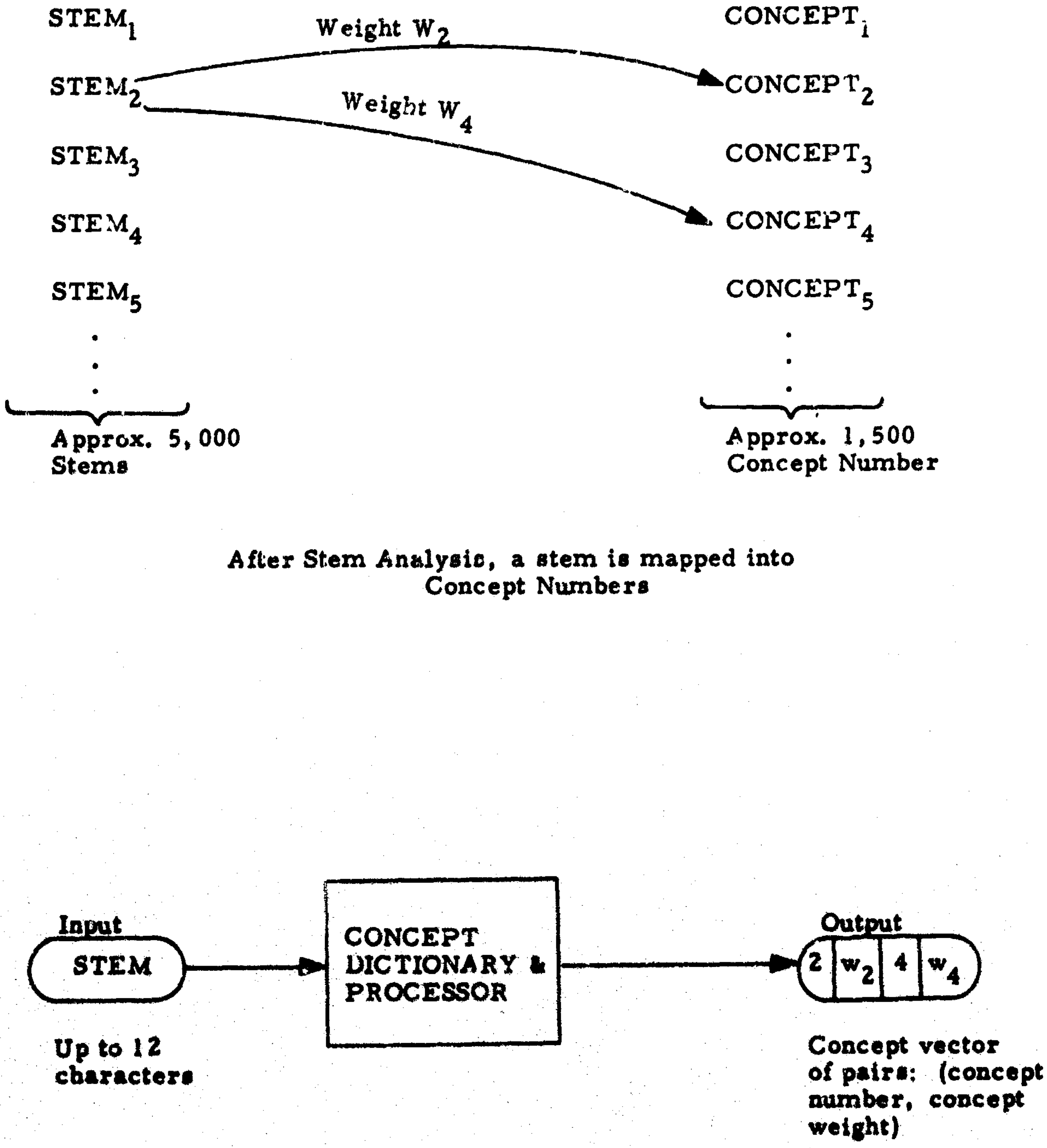
III. 2. 1. 4. 4 Impact on Design. Does the use of 3000 rather than 5000 documents make a significant difference? How many occurrences of a content stem are expected per document, on the average? If the number of occurrences and documents are small, it is possible to reduce the figure of 250, 000 words for the vectors to a figure allowing single pass in-core processing with a ranking table technique.

It is felt that some experimentation with an actual data base is required before any decision to replace the approach of Section III. 2. 1. 2 As that approach is known to be feasible, it is to be used unless an opportunity for experimentation occurs.

III. 2. 2 Concept Vector Generation--Dictionary Lookup

Figure III-8 shows schematically the function performed by the dictionary produced by the occurrence correlation process. There are about 5000 content stems, of which about 1500 are selected as key stem or concepts. The dictionary's purpose is to accept a content stem consisting of up to twelve characters and generate a short vector that relates the content stem to several key stems. The dictionary, therefore, performs a many-many mapping from the set of content stems to the set of key stems; and the mapping is a weighted one.

The programs have been designed as modular units. This allows for changing the correlation measure or the clustering schemes in the future. In use, the dictionary accepts free text words through the stem analyzer (III. 1. 2), which also deletes common words. The dictionary ignores noncontent stems.



After Stem Analysis, a stem is mapped into Concept Numbers

Figure III-8. Function Performed by the Dictionary Processor

Many schemes for dictionary lookup are available, including the symbol trees, key-to-address transformations, balanced trees and triplet searching algorithms compared by Lowe (29). Hays (21) gives an introduction to simple tree-structured dictionaries and binary searching dictionaries, while Brooks and Iverson present a detailed analysis of binary searching algorithms.

In the present case only about 5000 "content stems" exist, a number which can easily be loaded into core for searching. This factor indicates that simply binary searching will suffice --and when the dictionary does not need to be partitioned it is a very rapid method. Even when partitioning is required binary searching is worthy of consideration, as is indicated in Habit's report (20) of a system utilising a vocabulary of some 75,000 words.

III. 3 Correlation

In the On-Line Retrieval system, automatic retrieval will be based upon a measure of the similarity between concept vectors, for both query-document and document-document retrieval specifications, as describ d in III. 3. 1. Section III. 3. 2 discusses the basis for the selection of the cosine as the correlation measure.

III. 3. 1 Query-Document and Document-Document Retrieval Modes

The user of the system can request retrieval based on a plain text query, or based on some document in the data base. If he elects to use an ordinary query, then a concept vector, called the query vector, is formed from his query, using the dictionary lookup processor (and other programs). If he elects document-document correlation, the document's concept vector is used as the query vector. In either case, a correlation measure between concept vectors is required.

III. 3. 2 Correlation Measures

Salton (44) reviews several correlation measures. One is the simple vector dot product, and dissimilarities may be considered as well as similarities by adding the dot product of the vectors to the dot product of their complement. Tanimoto's measure and the cosine have the advantage of falling in a specified range (0 to 1). In much of the SMART work the cosine and overlap measures are used.

Euclidean distance between two vectors in the hyperspace is intuitively attractive as a correlation measure. However, consider the case in which two concept vectors $\overline{X}$ and $\overline{Y}$ are related by the equation

$$\overline{X} = a\,\overline{Y} .$$

In other words, the two concept vectors contain identical concept occurrences and weights, except for a scalar multiplier. It is desirable in this case that the correlation measure be 1; however, the Euclidean distance between these two vectors is given by

$$D = \sum_i (x_i - ax_i)^2$$

$$= (1 - a)^2 \sum_i x_i^2 .$$

Thus the Euclidean distance in this case does not give the desired value for the correlation measure. On the other hand, the cosine is given by

$$c = \frac{\sum_i ax_i^2}{\sqrt{\sum_i x_i^2 \sum_i ax_i^2}}$$

$$= 1.$$

In a broader sense, it is evident that the correlation measure should be a function of the angle in hyperspace between two concept vectors, rather than any sort of distance measurement, because the angle is determined by the relative magnitudes of the vector's components, rather than their absolute magnitudes. The cosine is an attractive measure of this angle because its value is 1 if the vectors are coincident and 0 if they are othogonal. Furthermore, the cosine function is well-behaved, and its value is always between 0 and 1. For these reasons, the cosine will be used as the correlation measure.

III. 4 File Organization

This section discusses the major characteristics of the file organization that has been selected for the on-line system. A survey of the various methods of file organization that were studied is presented, followed by a discussion of clustering techniques, and an evaluation of their applicability to this effort. The selected file organization is then presented, and compared to the desired characteristics for a clustering technique suggested by Ide et. al. (23) in ISR-12.

III. 4. 1 Comparison of Various File Organizations

This discussion covers some of the possible ways files for the on-line system could be organized. Here the problems of correlation algorithms and thesauri are neglected, and emphasis is placed on the identification of documents with concept vectors sufficiently "close" to a given concept vector, as measured by an unspecified matching algorithm. Linear file searching is immediately rejected for on-line use, and it rapidly becomes apparent that one of the critical problems is that of holding response time at a reasonable level.

It should be realized that the present problem differs greatly from the usual retrieval problem in which searches are based on relational operators and documents are characterized by the presence or absence of index terms (such as keywords). Systems for that problem have been developed and operate on-line using very large document collections. Here we are concerned with retrieval based on real document and query concept vectors. A query concept vector may be generated as shown in Figure III-8. Query concept vectors may also be derived from modification of previously generated vectors by the searcher, and document concept vectors may be used for query whenever document-document correlation is desired. Their origins affect the analysis below only in that document-document correlation can result in rather rich query vectors.

Some of the parameters to be considered are:

a_i The accession or internal sequence number of the i-th document; also, loosely used for the name of the i-th document. In many cases it may be possible and convenient to let $a_i = i$.

D The number of documents in the collection.

N The number of distinct concepts.

S The total number of occurrences of all concepts with nonzero weight in the characterization of the entire collection.

c_j The concept code for the j^{th} concept; also loosely used for the name of the j^{th} concept.

w_{ij} The weight of the j^{th} concept in the i^{th} document. Note that this is not the same w_{ij} used in III.1.

We have now progressed from the use of stem weights to the use of concept weights.

f(j) The total number of times the j^{th} concept* appears with nonzero weight in the entire collection. Therefore,

* NOTE: If the same stem occurs several times in a document's abstract, and that stem is uniquely mapped into a single concept, this is considered only one concept occurrence. It is expected that the weight of the concept would be higher if the stem occurred twice. Thus, a concept is either present or absent in a given document, and relative importance if present is indicated by its weight.

$$S = \sum_{j=1}^{N} f(j).$$

h(i) The number of concepts appearing in the i^{th} docun with nonzero weight. Therefore,

$$S = \sum_{i=1}^{D} h(i).$$

I The number of concepts appearing in a search request with nonzero weight.

x_i The value returned by the correlation algorithm upon comparison of the document vector of the i^{th} document with the present query vector.

G The number of logical record accesses required to service a query.

p(j) The probability that a concept in a search query is the j^{th} concept.

Perhaps some explanation of the meaning of G is in order. Electronic switching times for direct-access storage devices are generally negligible in comparison with other delays. In the presen case, the primary causes of delay are rotation of the disk and translatic of the heads. If two logical records have been unrelated in the building of the files and if they are referenced one after the other, translation of the heads will be required a certain (large) proportion of the time. Therefore, a quantity to be minimized is G, the number of logical reco accesses required to service a query. An additional consideration is the relationship between physical and logical records, as minimization of G does no good if the number of physical records needed to contain a logical record is greatly increased.

For many file organizations, it turns out that the rank-frequency distribution of concepts can greatly affect mean query response times. Another important factor is the distribution occurring in search queries. A recent paper(30) gives an analysis of these effects for some aspects of conventional keyword retrieval. The fact that these effects exist must be kept in mind during the present design, for they might influence performance of the concept-oriented system. As an example selected from that paper, Figure III-9 shows the ratio of retrieval time when query keyword distribution follows a slightly modified version of Zipf's law* to the time when the distribution is uniform over the vocabulary of keywords. The file organization in the case illustrated is that of linked lists, also mentioned below in connection with the present problem.

In the concept-oriented system different thesauri and content analysis algorithms are to be used, so very little can be said about the distributions. Some crude estimates are made here in order that initial evaluation of file designs may be performed.

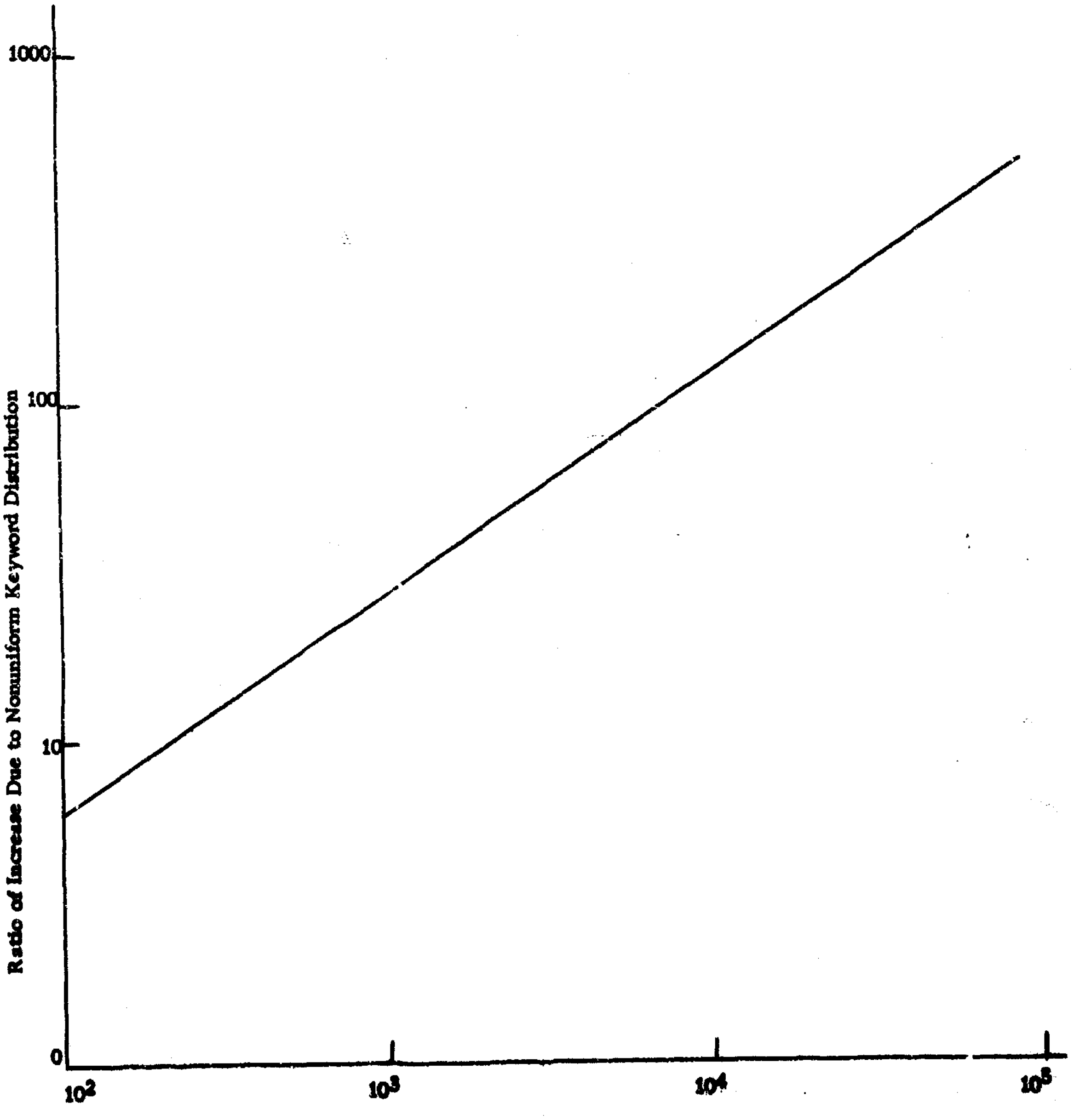
The number of concepts appearing in any document with nonzero weight is assumed to lie between five and fifty, as is the number of concepts in any search request.** The collection is assumed to consist of 40,000 documents. There are about 1500 distinct concepts. In terms of the previously defined symbols·

$$5 \leq h(i) \leq 50$$
$$5 \leq L \leq 50$$
$$D = 4 \times 10^4$$
$$N \cong 1500.$$

The total number of nonzero-weighted concept occurrences is the

* See subsection III.2.1.1.

** The upper limit would result from document-document correlations.



N (Number of Distinct Keywords)

Figure III-9. Effect of Nonuniform Keyword Distribution

sum of h(i) over all documents, so taking the extreme ranges of h(i):

$$2 \times 10^5 \leq S \leq 2 \times 10^6.$$

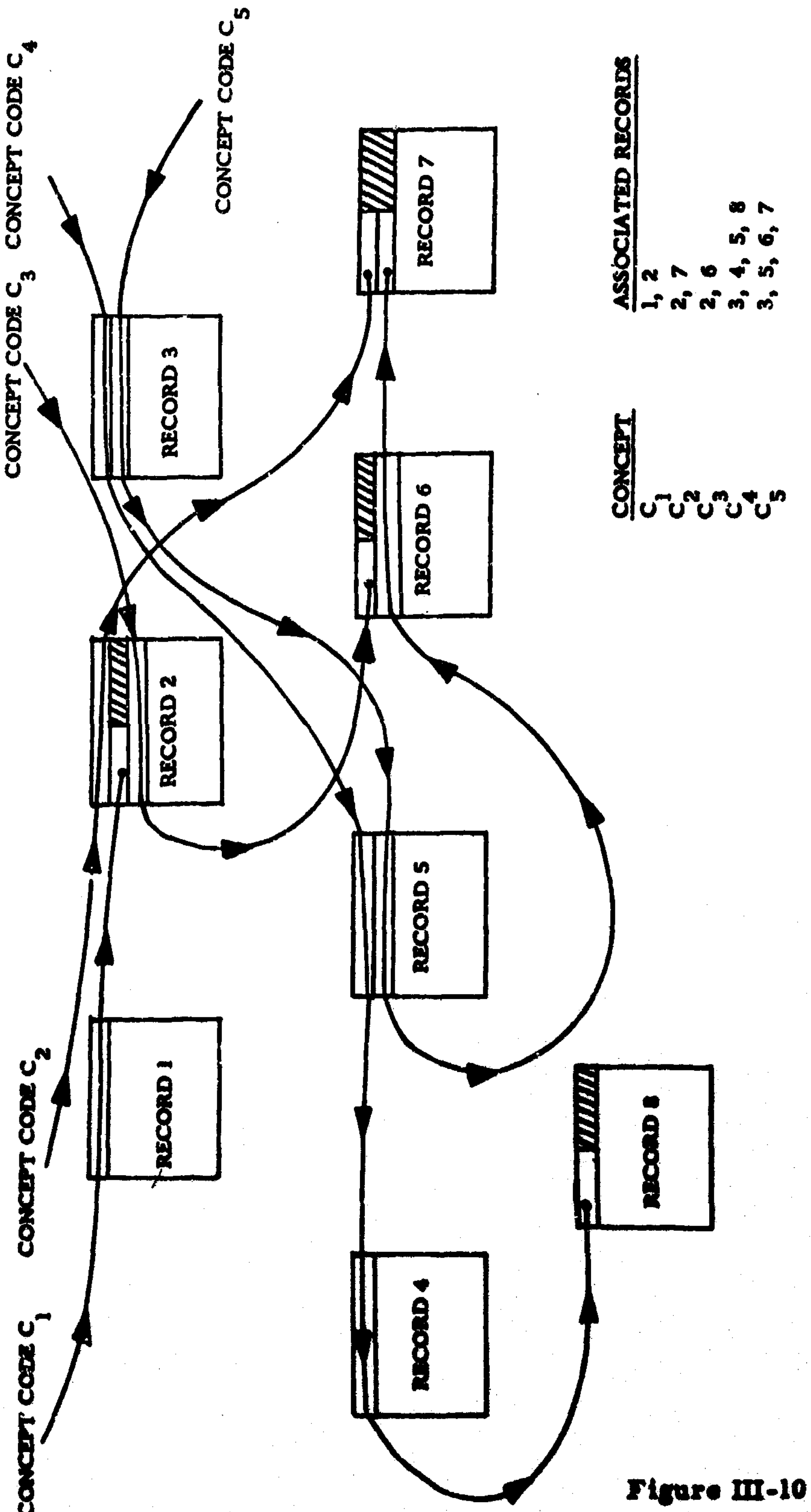
The usage of concepts in characterizing the file is unknown, but the average value is $\overline{f(j)} = S/N$.

The remainder of this section is devoted to the examination of several possible organizations.

III.4.1.1 Plan 1. In this plan a linked list structure is used. Each linkage chain corresponds to a concept and each logical record to a document. Figure III-10 shows this. The contents of a record are simply the concept vector for the document (all the c_j, w_{ij} pairs for which c_j has nonzero weight w_{ij} in document a_i). Record contents are shown in Figure III-11.

Figure III-12 schematically illustrates the method of retrieval. For every concept in the query, a thread (i.e., a linkage chain) is followed. Each threaded record corresponds to a document, and contains the document's concept vector. As soon as that vector is retrieved for a document, it and the query vector may be processed by the correlation routines, giving a pair (a_i, x_i) which is a measure of the relevance of document number a_i to the query. These pairs are entered in a table for eventual ranking of the retrieved documents.

Two refinements are possible. First, threads for different concepts of the search query may often intersect. In order to prevent repeated correlation of identical documents, either one of two methods may be employed. First, a list (of unknown length) of accession numbers previously checked may be kept in order that processing of a document concept vector is avoided if that document has already been encountered. Alternately, since the codes of all relevant concepts appear in the concept vector of the document, if one of these is the same as the concept naming a thread already processed then the present document may be skipped.



CONCEPT	ASSOCIATED RECORDS
C_1	1, 2
C_2	2, 7
C_3	2, 6
C_4	3, 4, 5, 8
C_5	3, 5, 6, 7

Figure III-10
Linked List Organization

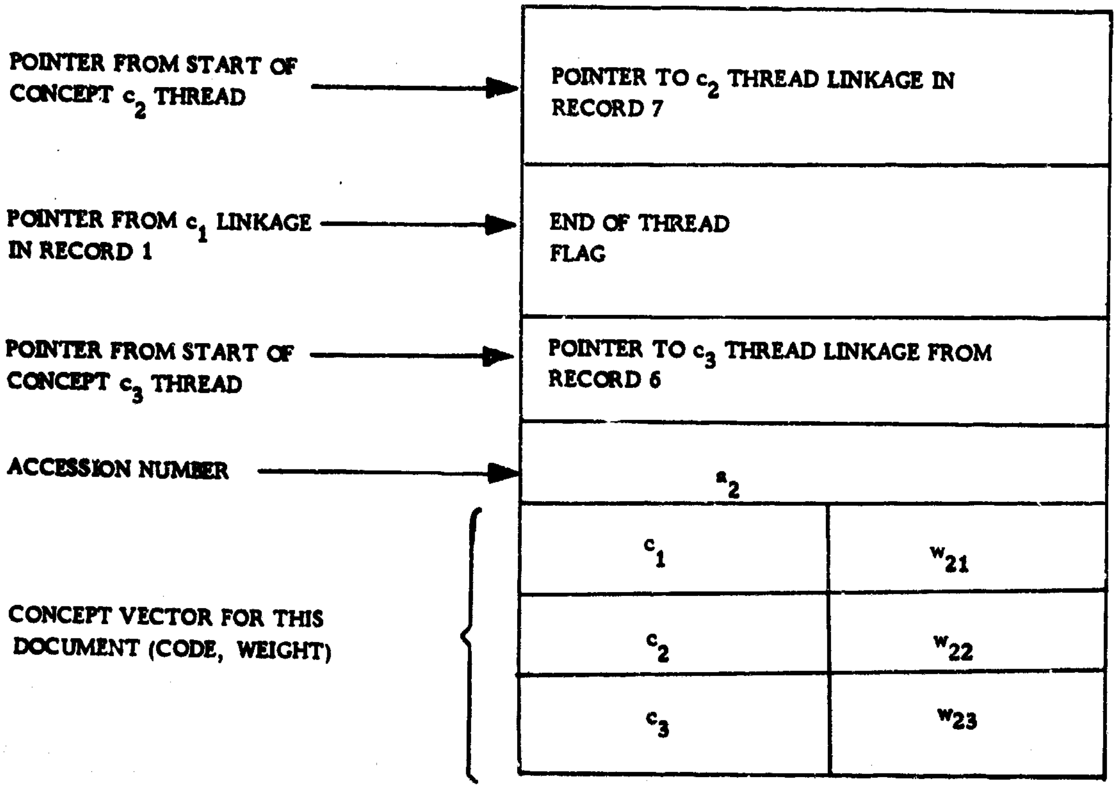

POINTER FROM START OF CONCEPT c_2 THREAD
POINTER TO c_2 THREAD LINKAGE IN RECORD 7
POINTER FROM c_1 LINKAGE IN RECORD 1
END OF THREAD FLAG
POINTER FROM START OF CONCEPT c_3 THREAD
POINTER TO c_3 THREAD LINKAGE FROM RECORD 6
ACCESSION NUMBER
a_2
CONCEPT VECTOR FOR THIS DOCUMENT (CODE, WEIGHT)
c_1
w_{21}
c_2
w_{22}
c_3
w_{23}

Figure III-11. Record 2--Expanded View

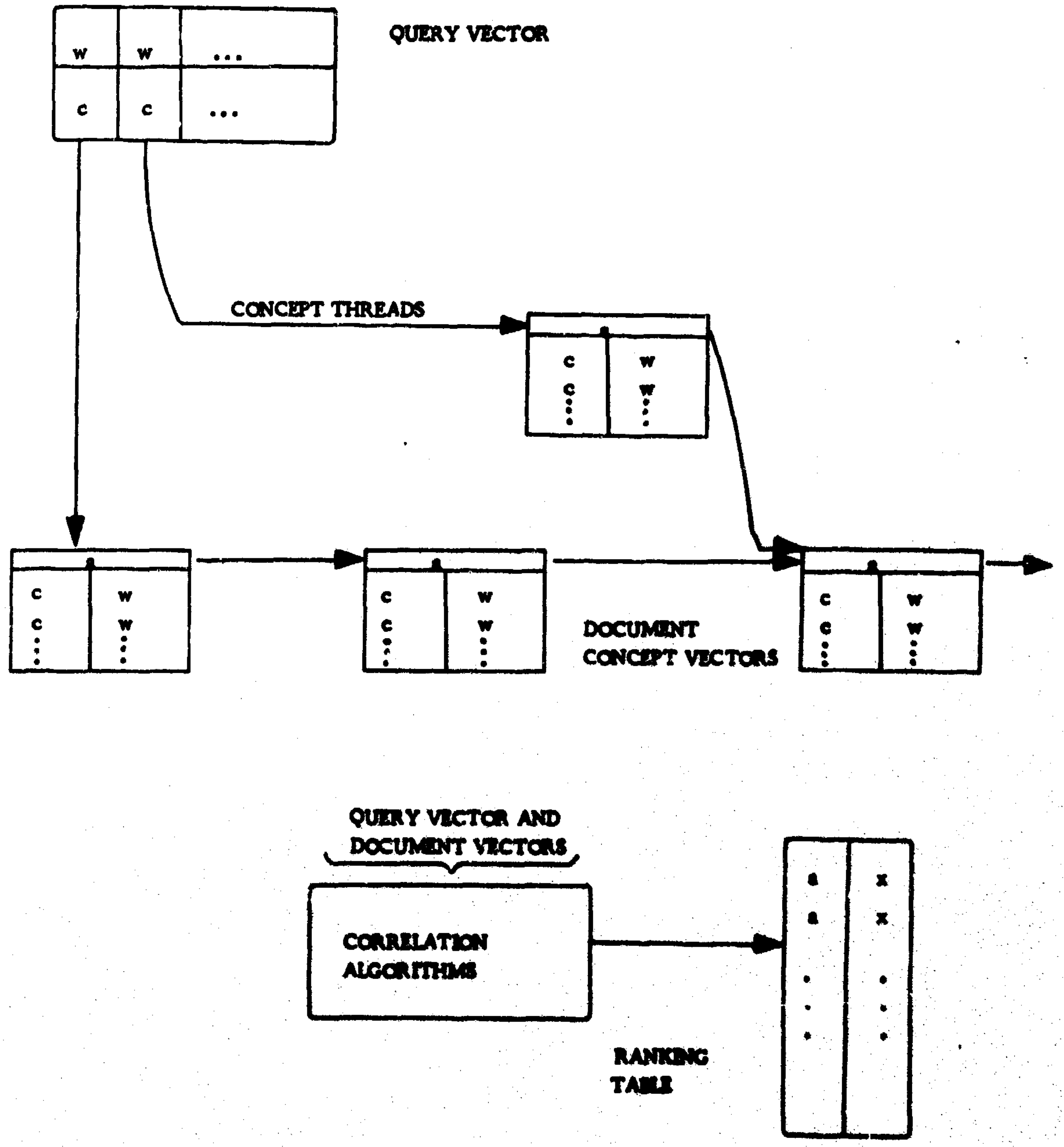


Figure III-12. Plan 1 --Threaded Lists

The second refinement avoids concern with a potentially large ranking table. One can determine a reasonable maximum number of documents for a request*, and so fix the table size. A pointer is assigned to the smallest table entry (i.e., the least relevant document), and then after the table becomes full new entries are added only if they are larger (in terms of relevance) than the present smallest entry.

Consider now the response expected from this organization. There are L threads to be followed, the thread named by c_j consisting of f(j) linked records. If it is assumed that a uniform distribution exists then f(j) = S/N, and so the number of logical record accesses is G = L(S/N). Using the estimated ranges of L, S, and N, the value of G lies between 7×10^2 and 7×10^4. Even with reasonably rapid disks, the lower number is rather high for effective operation.

(It should be mentioned that in this application one of the advantages of threaded lists has been lost. Generally, they find their application when the intersection of several threads is desired; then the search can be performed on the shortest list. In this case, however, it is possible for a document to possess a weight of zero on one of the query concepts, but still rank sufficiently high to be a desired "hit".)

III.4.1.2 Plan 2. This scheme makes use of a file of accession numbers, inverted by concepts. If concept c_j appears in document a_i with weight $w_{ij} > 0$, then the accession number a_i appears in the list with name c_j. A second file contains document concept vectors, named by their document's accession number. Figure III-13 illustrates this scheme.

In order to service a query, for every concept in the query that concept's inverted list is accessed. For every accession number in the inverted list, there exists a document vector named by the document's accession number. The document vectors may be applied one at a time to

* For system flexibility, this should be programmed parameterically.

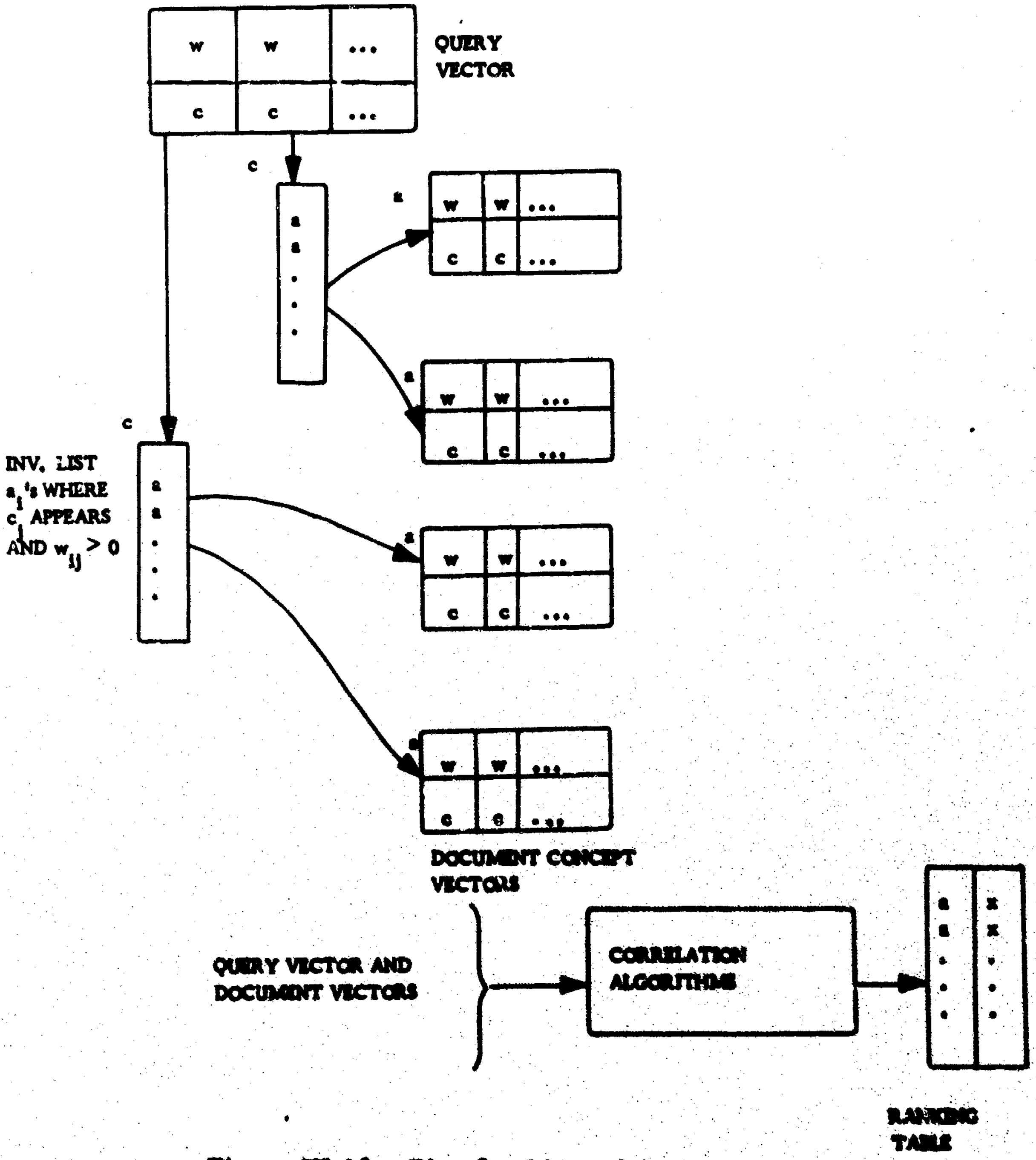


Figure III-13. Plan 2--Lists of Accession Numbers Inverted by Concept

the correlation algorithm, which produces a value x_i, to be entered with a_i into the ranking table. The ranking table overflow procedures mentioned for Plan 1 may be applied.

If the query concepts are $c_{r_1}, \ldots, c_{r_L}$, then the number of record accesses required is:

$$G = L + \sum_{j=1}^{L} f(r_j), \text{ and when } f(r_j) = S/N \text{ is assumed, then}$$

G = L (1 + S/N), a result even worse than that obtained previously.

III.4.1.3 Plan 3. This is essentially a refinement of Plan 2. In it the repeated accession of the same document vector is avoided. This can be accomplished either by forming the union of L retrieved inverted lists of accession numbers before retrieval of any document vectors, or by keeping a "checkoff" list of accession numbers already found and ignoring any accession numbers already on that list.

Two extremes in the number of logical record accesses required depend on the request vector. Either extreme is unlikely, but the actual effect will lie somewhere between them. In the first, the intersection of all the inverted lists named by the query concepts is null, and so no gain results. In the second, all the inverted lists are identical, so G = 1 + S/N. Still, the values of G lie between 2×10^2 and 2×10^4 for the lower extreme case and between 7×10^2 and 7×10^4 for the upper.

III.4.1.4 Plan 4. There is, however, an additional refinement. Unfortunately, it is difficult to estimate the gains that would result because of lack of information on the contents and classification of the documents, but the method can be outlined.

The inverted lists of document accession numbers must be stored so that the numbers are in order. In forming the union, the primary operation is merging a list just retrieved into the existing list, so that the generated list contains all the relevant accession numbers <u>in order</u>. A single document vector requires relatively little storage--so a physical record or "bucket" that may be accessed from disk at one time can contain several such vectors. Suppose that the vectors are stored in buckets in order of accession numbers. Then, as shown in Figure III-14, the union of inverted lists may be scanned in order and relevant <u>buckets</u> retrieved. Document vectors present in a retrieved bucket which correspond to accession numbers in the union inverted list may immediately be processed and assigned their x_i values; the other vectors are disregarded.

It can be seen that this approach can reduce the number of accesses required since a single-access picks up at least one document vector. Let an average of b vectors be contained in a bucket. The analysis begins like the classical occupancy problem*. If r balls are placed in n cells "at random"**, the probability that m cells are empty is $p_m(r, n)$. In the present case the number of cells is the number of buckets in the system: D/b. The number of relevant document vectors to be retrieved is r; the number of buckets containing no relevant document vectors is m. Therefore, n - m = D/b - m buckets must be retrieved, so

$$G = \sum_{n=1}^{n} (n - m) p_m (r, n),$$ the expected number of accesses

required to service a request. Substituting for the probability:

$$G = \sum_{m=1}^{n} (n-m) \binom{n}{m} \sum_{v=0}^{n-m} (-1)^v \binom{n-m}{v} \left(1 - \frac{m+v}{n}\right)^r .$$

* See reference (17), page 91.

** The probability of each arrangement is n^{-r}.

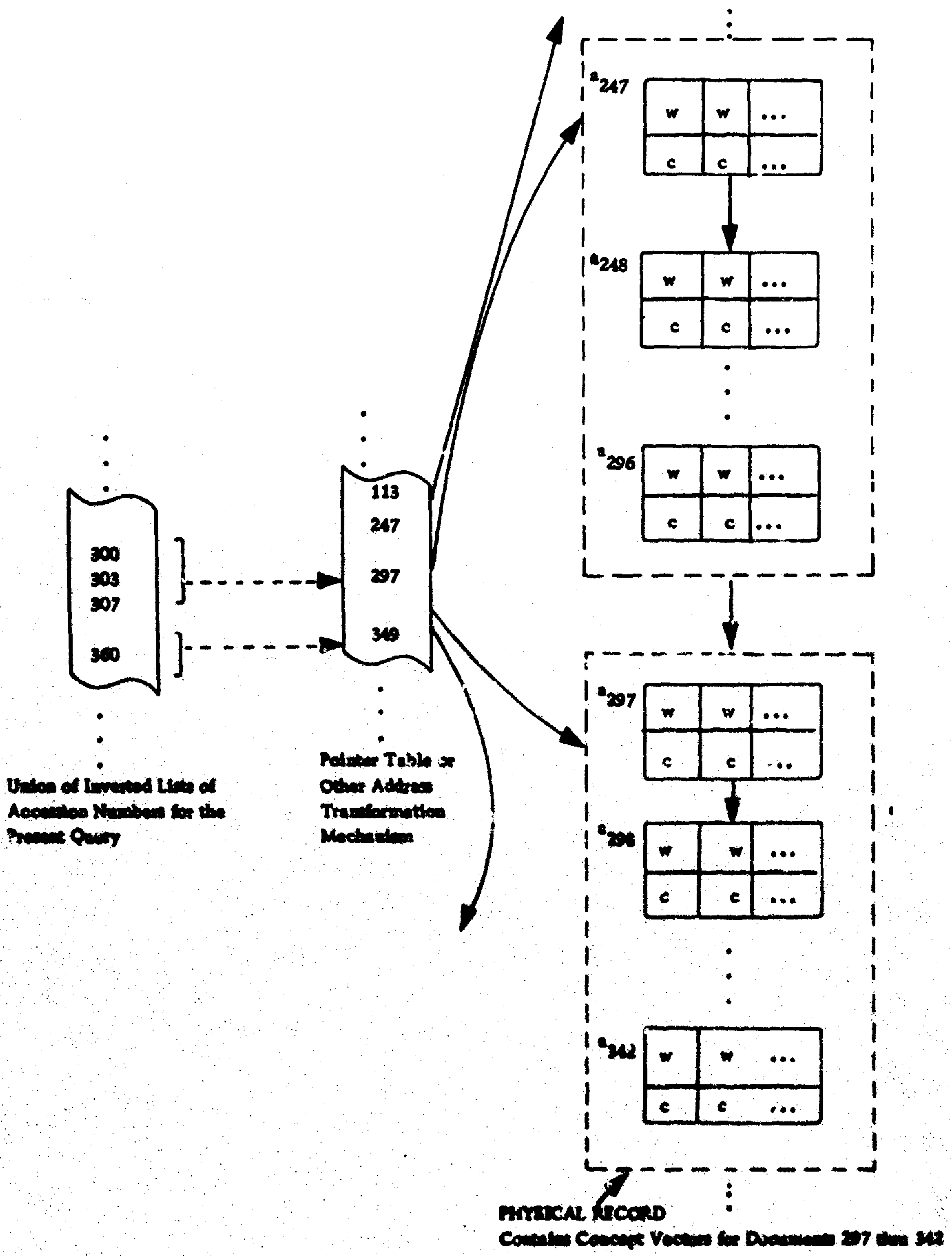

300
303
307
360
Union of Inverted Lists of Accession Numbers for the Present Query
113
247
297
349
Pointer Table or Other Address Transformation Mechanism
a247
w
c
...
a248
a296
a297
a298
a342
PHYSICAL RECORD
Contains Concept Vectors for Documents 297 thru 342

Figure III-14. Plan 4--Bucket Organization of Document Concept Vectors

When more information is known about the values of the variables, this analysis may be expanded. It is reasonable to believe that a Poisson distribution can be assumed, if accession numbers are arbitrarily assi ed to documents. But further improvements are possible if the assignmer is not arbitrary.

The term "accession number" has been used to designate son document identifier, and such a number may be internal to the system. Suppose that these numbers are assigned to documents after a prelimin y classification pass. Just as documents that are (in some sense) relatec are found adjacent in the one-dimensional space of library shelves, the concept vectors for these documents would <u>tend</u> to be close to one anoth and then similar documents would be grouped in the same buckets. Obviously, this would reduce G greatly. The mapping would not group all relevant documents for any request--if it could, concept vectors we d not be needed at all. But a good mapping (or concept vector transforma on) would help quite a bit. Two preliminary suggestions for this follow.

Using some measure of "distance" (cosine, for example), a packing penalty matrix (31) could be formed by the difference between th matrix consisting of all ones and the document-document distance matrix. Assignment of documents to buckets could then be performed by use of one of the recently developed packing algorithms (32).

A second approach would be to group documents by their conc nt of greatest weight. This would be much simpler to implement than the first attack mentioned above, and it might work as well.

III.4.1.5 <u>Plan 5.</u> As shown in Figure III-15, this organization has on inverted file for each concept. In every file there exists both the acces n number for every document in which the concept naming the file appears with nonzero weight, and the associated document vector. Obviously, only one file access is required for each concept in the query vector: G=

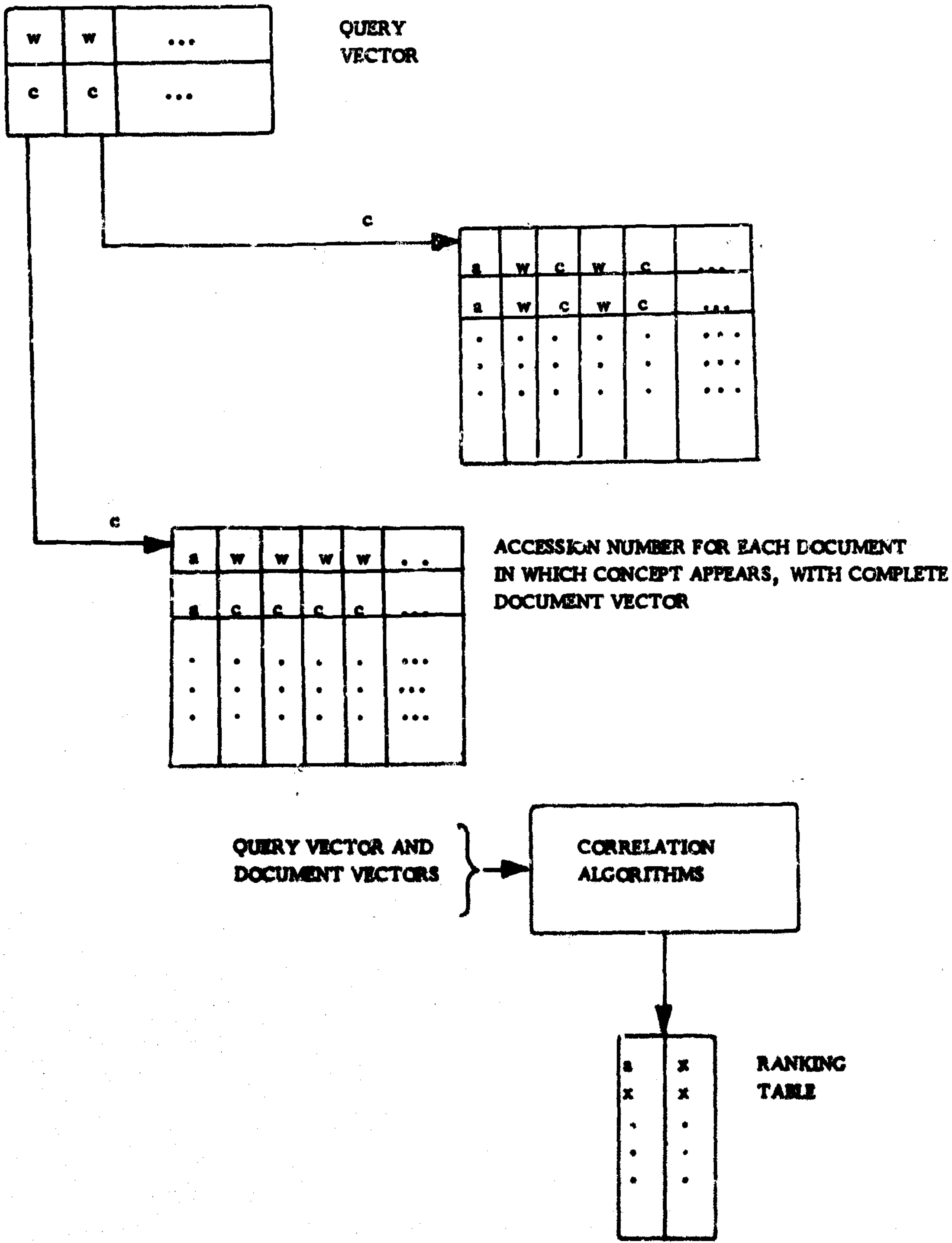


Figure III-15. Plan 5--All Files Inverted by Concept and Contain Complete Document Vectors

But as the inverted files are very large, many disk accesses may be required. For instance, assume f(j) = S/N and h(i) lies between five and fifty. This means that every inverted file contains between 2×10^2 and 2×10^5 accession numbers and concept vectors, and that every concept vector contains from five to fifty nonzero concept weights and the corresponding concept codes. Assumptions on packing of data into computer words indicate values on inverted file sizes to range between 10^3 and 10^7 words each, and there are as many files as concepts in the system.

It should be noted that for some applications, particularly when storage with Data Cell-like characteristics is employed, this scheme might be a good one. For instance, if the average concept appears in 2000 documents, and the average document is characterized by 17.5 concepts, inverted file size is 35,000 words.

III.4.1.6 Plan 6. This plan requires only L file accesses, and avoids large inverted files. However, it has other disadvantages (some of which are resolved in Plan 7).

Inverted files are named by concepts, as shown in Figure III-15. Each such file contains every accession number where the concept appears with nonzero weight, and the weight. So the jth such file is named c_j and contains f(j) pairs of accession number and weight. Such a pair could probably be packed into a word; therefore, a file contains between 2×10^2 and 2×10^4 words with 2000 a likely figure.

But the set of concept vectors to be matched with the query vector is built up by concept (column by column as shown in Figure III-16). Ranking must be done by document, in a row by row manner. This means that before any document may be rejected, the entire document-concept array must be constructed. This can be huge, and since it is constructed in a sequence different from that of the use of its components, it is desirable to keep it all in core at one time.

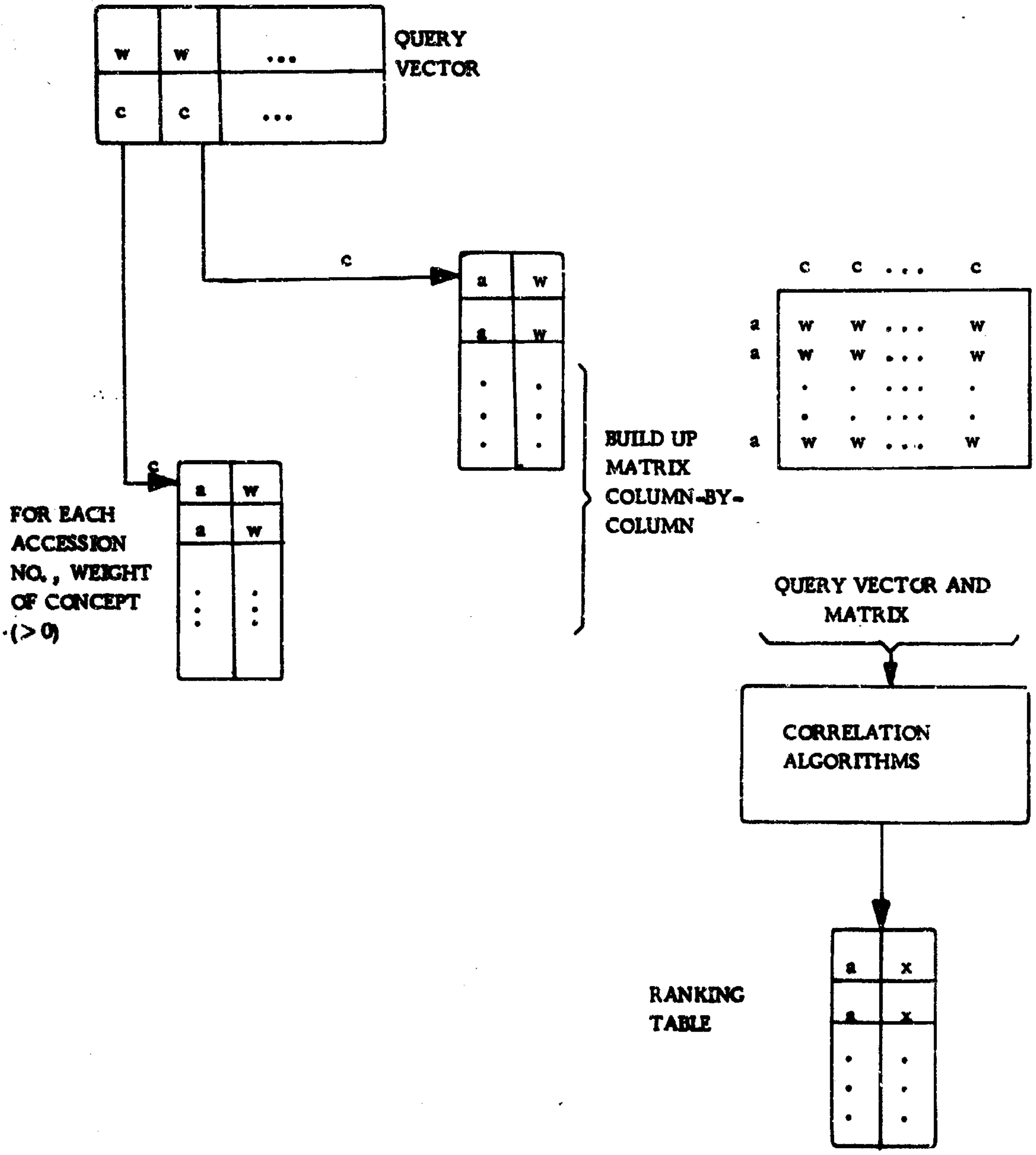


Figure III-16. Plan 6--Files Contain Complete Data on a Concept's Use in Collection

A method of overcoming these difficulties is discussed below.

III.4.1.7 Plan 7. This is a modification of Plan 6, in which difficulty of manipulation and storage limitations are evident. In Plans 1 through 5, it is possible to obtain correlation values for documents one at a time. This makes it possible to reject the less promising documents, and arrive at a list of the most highly ranked documents. To modify Plan 6 in order to achieve a similar effect, the inverted files used there may be partitioned into buckets. This is shown in Figure III-17.

Suppose that the first document-concept table is built up as in Plan 6, but considering only a_1 though a_{K-1}, where K is a parameter chosen as a function of available core space and disk blocking. Then complete data for assigning correlation values for these documents may be entered in a table no greater than K-by-L in size. The data in this table can be processed and the ranking table formation begun. This requires L file accesses. Next a_K through a_{2K-1} are considered, and the previous contents of the document-concept table can be destroyed. As this process continues, the relevant documents with lowest correlation values can be removed from the ranking table. An additional advantage to this method is that if a fixed size document-concept table is employed, then the entries can be identified by position alcne.

Plan 6 would result in a value of G = L file accesses, but since the inverted files can be quite long this might result in many more disk accesses. In Plan 7, a sweep must be made through the concept vector file for each partition of the set of accession numbers. But the buckets of inverted files retrieved may be made small enough to be retrieved in one disk access. Therefore, $G \leq LD/K$. "Holes" in the list of relevant accession numbers reduce G from its maximum value.

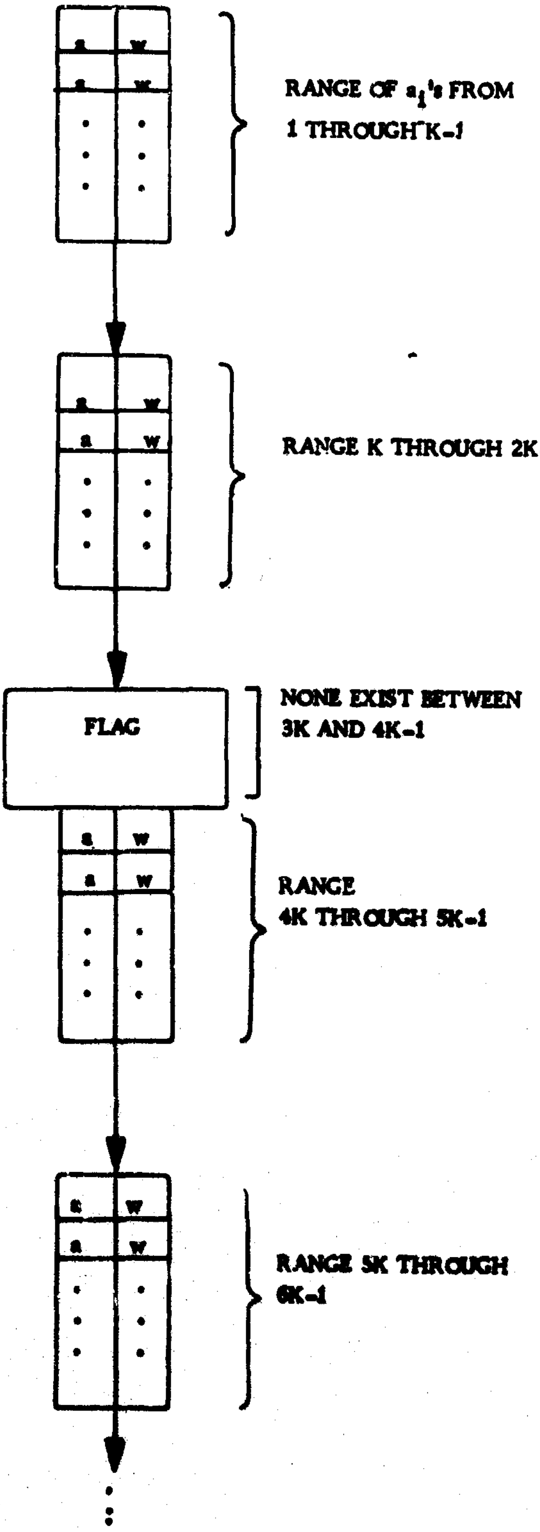


Figure III-17. Bucket Structure of Inverted Files for Plan 7

III. 4. 2 Document Clustering and File Structures for the On-Line System

This discussion treats document clustering techniques and their applicability to the file organization problems of the On-Line System.

The usefulness of automatic classification of document collections is widely recognized. Everyday experience yields many examples where clustering has been found useful in a variety of retrieval applications. For example, Doyle(15) points out that "Stores and supermarkets have things arranged in orderly fashion... newspapers have different kinds of news and advertisements bunched in certain sections." Even the stacks in a library, the manual information retrieval system most nearly resembling a mechanized information retrieval system, are organized by category to facilitate human searching of the stacks. Hints at this approach have been made in previous sections.

The extension of the idea of classification for ease of retrieval to mechanized information retrieval systems, then, is a natural one. Even for systems operating in the batch mode, motivation for document clustering has been found, as shown by Borko's (5) statement:

> "It is possible... to eliminate classification entirely and search the entire document file... However, ... this is an inefficient search strategy. The storage of a large collection of documents would require a number of reels of magnetic tape, and it would be time-consuming to serially search through all these tapes. Certainly it would be more efficient if one could be reasonably certain that the desired documents are all located in one place. This is precisely what classification is supposed to accomplish...".

The use of on-line information retrieval systems has intensified interest in document clustering. As pointed out by Lesser (27), a batch processing system can accumulate a large number of queries, and then search the whole document collection, processing all the queries at once. An on-line system, however, must process one query at a time; a complete search of the file for each query is not economical for large document collections. Lesser proposed the use of a two-level search, along with document clustering, to perform the retrieval process in on-line systems:

1) find the clusters whose centroids are most correlated with the query;

2) search the selected clusters for documents that fulfill the query.

The reasons for using document clustering in a mechanized information retrieval system, then, are two: the general argument that greater efficiency can be obtained by restricting the search size; and the argument based on the necessity for on-line systems to handle single queries. Because of the size of the data base to be used for this project, manual classification is impossible; therefore, the following discussion of classification techniques covers only automatic classification.

Early techniques for automatic document classification used a similarity matrix. For a collection of D documents, a D x D matrix was computed, whose entries were some measure of the similarity between two documents. The measures of similarity used were usually symmetric, so that about only $D^2/2$ entries were stored. In practice, the matrices were found to be only about 10% occupied, further reducing the number of entries stored to about $D^2/20$. In one experiment, a similarity matrix with 400,000 possible entries contained only about 48,000 and took about a minute to compute (25).

Unfortunately, the size of the similarity matrix grows with the square of the size of the document collection; the computer time required for the necessary computations also increases at least as rapidly. Maron (34) suggests a two-part clustering technique to avoid some of this undesirable growth. In Maron's technique, the documents are first preprocessed and assigned to a few macro categories; in later processing runs, each macro category is subdivided into categories.

Maron's technique undoubtedly makes automatic clustering feasible in some marginal applications; however, the problem of rapid growth of storage requirement and computer time requirement is merely postponed and not avoided.

Doyle (15)has suggested a clustering algorithm that greatly reduces the growth rate of storage and running time requirements as the size of the document collection increases. This algorithm completely avoids the computation of a similarity matrix. Starting with some initial classification scheme, some sort of concept centroid is computed for each cluster. Once this is done, each document in the collection is scored against the centroid of each cluster; a new clustering is then formed, with each document placed in the cluster whose centroid correlates most highly with the document's concept vector. The process is repeated until two successive iterations produce identical clusterings. The computer time required for this process varies as $D \log_r D$, where r is approximately ten.* Core storage requirements vary linearly with the size of the document collection and the number of clusters to be formed.

Doyle's algorithm appears to be an important advance in the state-of-the-art of automatic document classification. However, it is not prudent to apply Doyle's clustering algorithm indiscriminately simply because it is an improvement over previous methods; the costs associated with document clustering are still very high, and should not be borne unnecessarily. Suppose a collection of one thousand documents can be clustered in one hour. Then Doyle's algorithm requires thirteen hours to cluster a collection of ten thousand documents, and eighty-three days for a million-document collection (13). Although eight-three days of computer time might seem a

* r is equal to the number of clusters formed, if this number is held constant throughout the process. Using the readily derived relationship $\frac{\log_a x}{\log_b x} = \frac{\ln b}{\ln a}$,

$D \log_r D = K D \ln D$, where $K = \frac{1}{\ln r}$.

bargain when compared to the 120 years that would be needed to perform the same task using the similarity matrix approach, usage of such a large amount of computer time clearly must not be undertaken unnecessarily.

A further difficulty in the use of Doyle's algorithm relates to its performance. It has been found that the final clusters produced by Doyle's algorithm are strongly dependent on the initial clustering arrangement. In fact, it has been suggested that for best results, the initial document clusters should be generated manually! (6) The design requirements for the On-Line System preclude this.

It seems reasonable to summarize the current status of automatic document clustering by stating that methods exist that can give satisfactory results if one is willing to pay the price in terms of computer time, programming complexity, and possible difficulties if the initial clusters are not sufficiently good.

The stage is now set for consideration of the clustering requirements for this project. The On-Line System will have not one, but three distinct files to which it will have access; concept vectors, bibliographic data, and the document collection itself. The data have been separated in this manner to permit different file structures and storage devices of different speeds to be used for the three files, because of their widely different access requirements.

The roles played by these three files are best illustrated by an examination of the sequence of operations that would take place while a user exercised the system. The user types in his query; the system then forms a query vector. The system retrieves from the concept vector file all possibly relevant document concept vectors, and compares them with the query vector, ranking them according to their correlation with the query vector. The user then has the option of printing various combinations of accession numbers, titles, authors, and actual documents.

Presumably, he would browse through the bibliographic information, then perhaps direct one or more abstracts to be printed either at his remote station or at the central computer facility, and then, using this information, reformulate his query in some manner. This process continues until the user obtains the information he seeks. The flowchart of Figure III-18 illustrates the major elements of this process.

The access requirements for each file are determined by the way in which it is used. The bibliographic and document files are discussed first, because they are the most straightforward, and then the concept vector file is considered.

The bibliographic and document files are not used by the system in processing a search request; therefore, their access requirements are determined solely by the requirement to avoid excessive delays for the user of the system. From the user's point of view, the ideal system response to a multiple-document request from the bibliographic or document file would be achieved if printing of the requested information proceeded without interruption once it began. Thus, at a printing rate of ten characters per second, this means that the maximum acceptable access times for the bibliographic and document files are about one second and one minute, respectively. Of course, it would be desirable to be able to retrieve the first document to be printed in less than one minute, if this can be done at a reasonable cost.

The access requirements for the bibliographic and document files, then, are not difficult to satisfy. The concept vector file, however, has very different access time requirements, that pose a far more difficult problem in system design. It is not sufficient to set some arbitrary standard for the retrieval of any single vector; the requirement must instead be stated in terms of groups of vectors. To illustrate this, suppose it is desired to have the system respond to a query within ten seconds. Suppose also that the worst case processing of a query may necessitate the comparison of a query vector with three thousand concept vectors. If the access

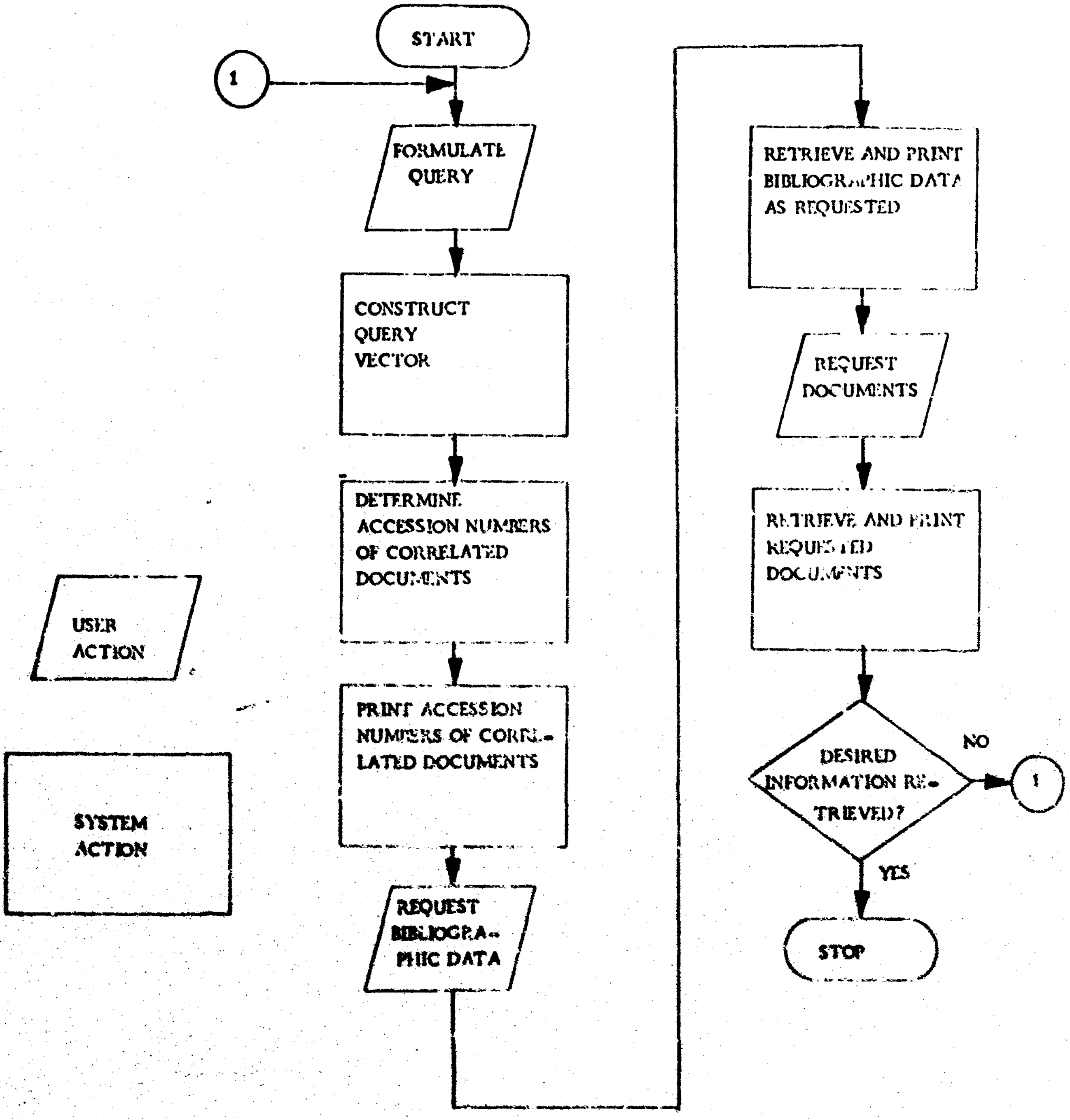

START
1
FORMULATE QUERY
CONSTRUCT QUERY VECTOR
DETERMINE ACCESSION NUMBERS OF CORRELATED DOCUMENTS
PRINT ACCESSION NUMBERS OF CORRELATED DOCUMENTS
REQUEST BIBLIOGRAPHIC DATA
RETRIEVE AND PRINT BIBLIOGRAPHIC DATA AS REQUESTED
REQUEST DOCUMENTS
RETRIEVE AND PRINT REQUESTED DOCUMENTS
DESIRED INFORMATION RETRIEVED?
NO
1
YES
STOP
USER ACTION
SYSTEM ACTION

Figure III-18. Major Elements of User-System Interactive Search

requirement is expressed in terms of single vectors, the average acces time for any vector must be less than three milliseconds! Clearly this is unachievable. If the concept vectors to be processed are retrieved ir ten groups, however, the average access time per group becomes one second. This goal is more reasonable. The only problem remaining is the selection of a file organization to guarantee that all relevant concept vectors will appear within a small number of groups.

III. 4. 3 Selected File Organization

The different access time requirements for the three files that make up the on-line system suggest that the three files might be located at three different levels of a hierarchical storage system. The concept vector file must be stored in the quickest-access device availab or else replicated on a slow-access device; the bibliographic data can b stored in a slower storage device, and the document file can be stored i an even slower device. Thus, if the on-line system were to be impleme ted on a computer system with a number of different levels of auxiliary storage, cost savings could be achieved by storing the massive data bas on a slower-speed device than the one used for the concept vector file.

Unfortunately, the peripheral complement of the computer requires that all three files be stored on disk. The access speed require nts must be met by file structure design alone. The bibliographic and docu nt files can be organized on disk by accession number; access to any reco will be within the requirements.

It appears at first glance that the best way to organize the concept vector file is to use one of the document clustering algorithms discussed above to place related concept vectors close to one another. Unfortunately, however, these clustering algorithms do not guarantee tl all concept vectors that may be relevant to a given query vector will be located in a reasonable number of clusters. If a query vector is suffici tly uncorrelated with every cluster center, the concept vectors that are highly correlated with the query vector will be distributed throughout a large number of clusters, requiring a large number of file accesses. This possibility is most undesirable. Even more undesirable, however, is the possibility that the system may fail to retrieve a relevant document in this situtation. Therefore, this method of organizing the concept vector file must be rejected.

This rejection of the use of a clustering algorithm for the concept vector file does not imply a criticism of the use of these techniques for their intended applications; rather, it is simply a recognition that the concept vector file is not similar enough to these applications to use the same file organization that is excellent for the application of a heuristic retrieval technique, that will retrieve most of the desired documents within a short time, most of the time. This application, however, requires an algorithmic retrieval technique; every concept vector that may be highly correlated with the query vector must always be retrieved within a specified time. A file organization that achieves these goals has been developed.

The organization recommended for the concept vector file fulfills the dual requirements of rapid and exhaustive retrieval of all desired concept vectors. The selected organization is a modification of Plan 5 of Section III. 5. 1. There will be one inverted file for each concept; the concept naming the file appears with nonzero weight, and the concept vector. Only one file access will be required for each concept in the query vector. If the average number of nonzero concepts per concept vector is j, the file of concept vectors will contain N = jD entries, where D is the number of documents in the file. For j = 50 and D = 40, 000, N = 2, 000, 000; if each record is stored in twenty words, then 40, 000, 000 words of random-access storage will be required for the concept vector file.

As mentioned in Section III. 5. 1, Plan 5 has two major drawbacks; each inverted file may require several disk accesses to retrieve it, and the storage space occupied by the entire file is rather large. This organization is obviously best suited for use with a Data Cell or similar storage device; moreover, since a Data Cell is normally required for the economical operation of a large data base information storage and retrieval system, it is not unreasonable to use a small portion of the Data Cell's storage capacity for indexing information.

The RADC computer is not equipped with a Data Cell type device. This can be overcome by simulating the desired file structure using a linked list. Each inverted list will be simulated by one linked list; therefore, only one copy of each concept vector will be stored on disk. Clearly, a sacrifice of execution efficiency has been made in order to adapt the file structure to available hardware; however, even this arrangement will permit an evaluation of the basic ideas that form the basis for the design of this system.

Section III. 4. 2 suggests file structures for the on-line system, contrasting the organization suggested for the concept vector file with previous clustering techniques. It is also useful, however, to evaluate the suggested organization's performance as a clustering method. Further, because the on-line system can be used for experimentation with various clustering methods, the suitability of the recommended file structure as a test bed also must be considered.

To compare the concept vector file structure with previous clustering techniques, a look at the objectives of previous work is useful. Typical is the discussion by Ide et. al. (23) in ISR-12:

> "if 100,000 documents could be usefully grouped into 1000 groups of 2000 documents each, . . . , only about 3000 comparisons, as opposed to 100,000, would be needed to find most of the documents relevant to a given query."

This statement implies that the clustered document collection will be about twenty times the size of the original, or that an average of twenty copies of each item in the collection will be made.

How does the concept vector file structure recommended above compare with these goals? The following analysis enables a comparison to be made. The notation used here is that of Section III. 4. 2, with the addition of a few parameters.

$r(i)$ the number of copies of the concept vector of the i^{th} document in the clustered concept vector file.

B the number of words of storage in one physical record.

E the number of words of mass storage occupied by the concept vector file.

P the number of concept-weight pairs per word.

$k(j)$ the numberof physical records required to contain the inverted file j.

$t(\ell)$ the number of physical accesses needed to process a request of ℓ concepts.

$g(\ell)$ the number of correlations needed to process a request of ℓ concepts.

V_i a concept vector; that is, a set of concept-weight pairs.

In the file structure suggested above, the clustered file will consist of N lists inverted by concepts.* Thus, each inverted list is associated with a unique concept, and can be named by the c_j associated with that concept. The number of entries in inverted list c_j is equal to f(j), the number of times the j^{th} concept occurs in the entire collection. Thus, the concept vector file will contain one concept vector for each concept occurrence in the original concept vector collection. Thus, the number of concept vectors in the file is given by

$$S = \sum_{j=1}^{N} f(j).$$

The number of copies of each concept vector in the file is given by

$$r(i) = h(i),$$

* To distinguish between the entire file and the group of all concept vectors containing a given concept, the terms "file" and "list" are used, respectively.

so the average number of copies of each concept vector is

$$\overline{r(i)} = \frac{\sum_{j=1}^{N} f(i)}{N} = \frac{S}{N} = \overline{f(j)}.$$

The size of the concept vector file can now be evaluated. The size is given by

$$E = \frac{\overline{r(i)}\ D\ \overline{f(j)}}{P}$$

$$= \frac{\overline{f(j)}^2\ D}{P} = \frac{S^2 D}{N^2 P} \quad .$$

Estimated values for P and D of 2 and 40,000 respectively, can be given with some confidence. It is more difficult, however, to estimate $\overline{f(j)}$ without additional experience with the data base; 30 has been selected on purely intuitive basis. This would mean that the concept vector file would occupy 18,000,000 words of mass storage.* Note, however, that the uncertainty in the estimate of $\overline{f(j)}$ makes the estimate of E very uncertain, because E varies as the square of $\overline{f(j)}$.

Now that the size of the file has parametrically been determined, what is its performance? More specifically, how many physical accesses to the file are required to process a query? Inverted file c_j will contain f(j) concept vectors. Thus**

$$k(j) = \left\lceil \frac{\pi(j)}{B\ P} \right\rceil + 1$$

where $\pi(j) = \sum_{i=1}^{D} \beta(i, j)\ h(i)$ and $\beta(i, j) = \begin{cases} 1 \Leftrightarrow c_j,\ w_j \in V_i \ \&\ w_j > 0 \\ 0 \text{ otherwise.} \end{cases}$

* This can be reduced by simulating the inverted lists with linked lists.

** Note "$\lceil x \rceil$" is used to denote "the smallest integer not less than x".

To compute the average for k(j), the average probability that $\beta(i, j) = 1$ for some i is given by $\frac{h(i)}{N}$. In the average, then, $\pi(j)$ becomes

$$\overline{\pi(j)} = \frac{\overline{h(i)}^2 D}{N} ,$$

and therefore

$$k(j) = \left\lceil \frac{\overline{h(i)}^2}{BPN} \right\rceil .$$

Now it is possible to obtain numerical estimates for $\overline{k (j)}$. Using 100, 2, and 1500 for B, P, and N respectively, the estimate for $\overline{k(j)}$ is

$$\overline{k(j)} = 1 ,$$

which merely states that most of the inverted lists will fit into one physical record of 100 words. This means that the number of physical accesses needed to process a query of ℓ concepts is approximated by, on the average,

$$\overline{t(\ell)} = \ell \; *.$$

Now that the necessary estimates have been found, the characteristics of this file structure can be compared to the general goals outlined by Ide et. al.(23) in ISR-12. Figure III-19 summarizes the overall characteristics of the technique suggested by ISR-12 and this report. The ratio of file size to original collection size is similar for both methods, as is the number of groups. The number of comparisons to service a query is much lower for this organization, but the meaning of this comparison is not clear because ISR-12 does not give the assumptions on which their computations were based. In general, then, the proposed file structure has somewhat more groups and makes somewhat more copies of each concept vector than might be ideal for a clustering algorithm; however, it is probably

* This is an approximation because there will be some inverted lists that will not fit into one physical record; meaningful estimation of the number of such lists is not possible at this time.

	ISR-12	III. 5. 2
File Size / Collection Size	20	30
Number of Clusters	1000	1500
Number of Comparisons to Service a Request	3000	300
Collection Size, Document	100,000	40,000

Figure III-19. Comparison of Clustering Techniques

more efficient in terms of the number of comparisons that must be performed.

It is not surprising that the overall performance characteristics of this structure are qualitatively similar to what might be achieved with a clustering algorithm. For although this file structure appears to be a simple inverted list organization, and superficially resembles coordinate indexing, in reality the information developed by the occurrence correlation process has been used to organize the collection into groups. This is an economical way to perform document clustering; concept identification and document clustering are conceptually identical processes. Because both processes are quite lengthy, it is worthwhile to perform both operations as one process.

The suggested file organization must also be useful as a portion of a system that is itself a test bed. Thus, this particular file organization must have some usefulness itself for testing purposes, as well as permitting other file structures to be tested.

This file structure has one main characteristic that distinguishes it from nearly every other structure; every concept vector that could possibly have nonzero correlation with the query vector will always be accessed. Thus, this organisation can be used as a standard to test other structures. Recall has been tested in the past by obtaining all relevant documents through painstaking manual searching of the data base; in this case, changes in recall due to variations in the file structures can be measured by running the on-line system with this structure, and then reloading the file with the test structure, and inserting the same queries.

The system's compatibility with various file structures is a programming detail and not a file structure characteristic. This structure should, of course, be set up with a centroid for each cluster; the centroid will be a concept vector with all entries sero except for the concept naming that cluster. The system will retrieve by first scanning the centroids; only the centroids of the concepts in the query will correlate at all with the query. In this manner, the inverted list structure will be compatible with other clustering methods.

SECTION IV

MAN-MACHINE DIALOGUE

The heart of the On-Line System is the software module which governs communications between the user and the system. This module--the dialogue processor-- also performs the executive function of the On-Line System. It calls on the routines which perform stem analysis, retrieval, ranking, dictionary lookup and all the other functions. It solicits queries and commands from the remote user, causes search queries to be executed, and reports and stores the results and generally leads the user through the array of tools available to him in his searching of the data base. The dialogue processor is, therefore, a communications package, a training aid, a file building program and an executive program all in one.

IV.1 General Design

The dialogue processor is designed, insofar as its functional characteristics appear to the user, with the overriding concept that different users of differing ability, needs, familiarity and goals will at various times attempt to use the system. In order for these attempts to succeed, the system must be geared to the user. The experienced user will not tolerate the delays incurred as lengthy tutorial messages are printed at the Teletype terminal; the inexperienced user will flounder without them. The inexperienced user wants to be led through the operation of the system; he does not, however, wish to be asked questions about optional employment of system functions with which he is not familiar. On the other hand, the experienced user wants to be able to marshal every last resource of the system. Finally, the inexperienced user should not be kept in a cocoon forever, and he must be at least given the opportunity to obtain an explanation of the various available features of the system.

IV.1.1 Flowchart Notation

The flowcharts in this Chapter are somewhat lengthy, so the following convention has been made in order that the reader may find the location of remote connectors. Connector names are of the form "xx-nn", where the "xx" portion is a numeric connector designator for the minor entry points or a mnemonic designator for major portions of the dialogue processor. The sheet of flowchart on which the entry point is located is sheet number "nn".

Conventional flowchart symbols are used with one exception. For brevity, when the user is asked a question which may be answered "yes" or "no", the input/output symbol also indicates the resulting branch in control flow; e.g.;

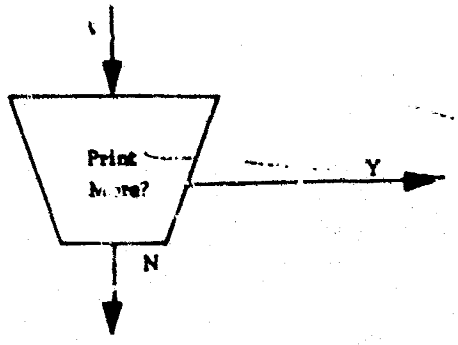


Figure IV-1. Input/Output/Branch Combination

IV.2 Functional Description of the Basic Dialogue Processor

The fundamental method of operation is embodied in the concept of a query sequence. Initially, the user sets up a retrieval command based on words. He is then given the opportunity to inspect the results of the retrieval, to modify the query or to discontinue the query sequence. During such a query sequence, a file of retrieved documents is built up. The three basic options available to the inexperienced user are:

"END" Terminate this search query sequence in order to start a new sequence or sign off.

"MOD" Modify or replace the present query and continu the present query sequence.

"DOC" Print data for documents retrieved during this sequence, or any documents of known accession number. The user is given a choice of the data to be printed.

Ten other options exist, and some of these are actually enter automatically for the inexperienced user.

While building the temporary file, the user can delete irrelev t documents. Since the file is built up by the process of executing differe retrieval requests, the re-retrieval of documents already retrieved onc during the sequence may be inhibited at the user's choice.

If bibliographic data for a document have been printed once during a single query sequence, it is unlikely that the user will want the data printed again. Such printing is inhibited, but the user (even the inexperienced user) can override this inhibition.

In addition to the various options available, there are several modes of operation. An example is the "terse" mode, in which message printed for the user are in an abbreviated form. The inexperienced user operates in "normal" mode, and need not concern himself with the other available modes.

IV.2.1 The Temporary File

Every time a retrieval is successfully executed during a quer sequence, information concerning the documents retrieved is added to the temporary file. The highest-ranked documents are placed first, continuing until all retrieved documents have been placed in the file or the file is full. The file capacity is 50 documents. Before a retrieval is executed, the user is informed that the file is presently empty, or informed of the remaining space and asked if additional space is requir or told that the file is full and that additional space must be created.

During any query sequence, each retrieved document is assigned a temporary identification number. This number is used only for convenience, since it is much shorter than the document's accession number. The user may need to specify a document for deletion from the temporary file, for the printing of bibliographic data or of the document itself, or for document-document correlation.

The temporary file contains only the following information:

1) Accession number;
2) Temporary identification number;
3) Flag indicating if the last executed retrieval retrieved the document;
4) Flag indicating if the bibliographic data for the document have been printed and the printing inhibition not removed;
5) Correlation obtained during the last retrieval of the document;
6) Rank obtained during the last retrieval of the document.

In addition to the temporary file, there is a list of documents whose retrieval is excluded. These are documents which have been retrieved at least once during the retrieval process, that the user does not want to re-retrieve.

IV.2.2 The Query Types

Initially, a set of query words is entered by the user. A file containing these words, their stems and weighted mapping into concepts is established. For additional retrievals during the query sequence, the file may be cleared and a new query entered. Or words may be deleted, added or replicated, building on the initial query.

After a retrieval, the query concept vector is retained. If the next retrieval is based on query words, the query concept vector is simply cleared and a new vector constructed from the query word file. In the case of document-document correlation, the user may either build on the existing query concept vector or generate an entirely new one.

It is also possible for the user to manipulate the query concept vector directly.

IV. 2. 3 Levels of Document Information

Information concerning documents is available on three levels. First is the temporary file information, obviously available only for documents retrieved during the present query sequence. The only permanent information in the file is the document's accession number.

There are also the bibliographic data, with such elements as author, title, date, etc. These data may be printed in a relatively short time, and the user may obtain them for either documents in the temporary file or for any other document whose accession number is known.

Finally, there are the documents themselves. These can be obtained in the same manner as the bibliographic data, and, of course, are comparatively lengthy. (In the presently contemplated data base, the "documents" are in fact abstracts of other documents.)

IV. 3 Operation of the System by the Inexperienced User

The flowchart TYRO (Figure IV-2) indicates functionally how the man-machine dialogue would appear to a user who uses only the options "MOD", "LOC", and "END", and operates only in normal mode. That is also described in the text below.

When the user first enters the on-line system, he is asked if normal operation is desired. An answer of "NO" results in a request for mode flag settings and an option selection, but here it is assumed that normal operation is indeed wanted. The user is then directed to enter initial query words.

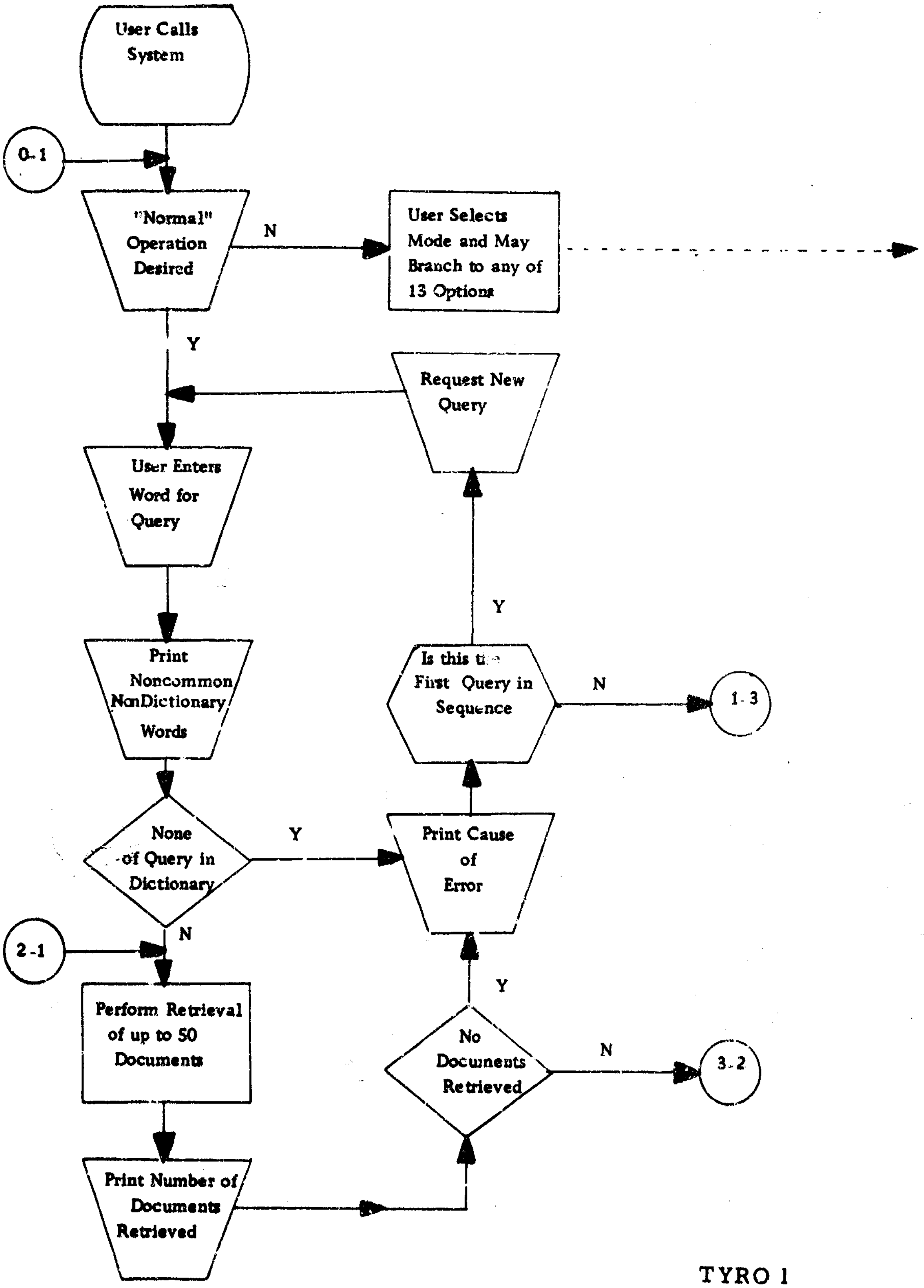

User Calls System
0-1
"Normal" Operation Desired
N
User Selects Mode and May Branch to any of 13 Options
Y
Request New Query
User Enters Word for Query
Y
Is this the First Query in Sequence
N
1-3
Print Noncommon NonDictionary Words
None of Query in Dictionary
Y
Print Cause of Error
N
2-1
Y
Perform Retrieval of up to 50 Documents
No Documents Retrieved
N
3-2
Print Number of Documents Retrieved
TYRO 1

Figure IV-2. TYRO

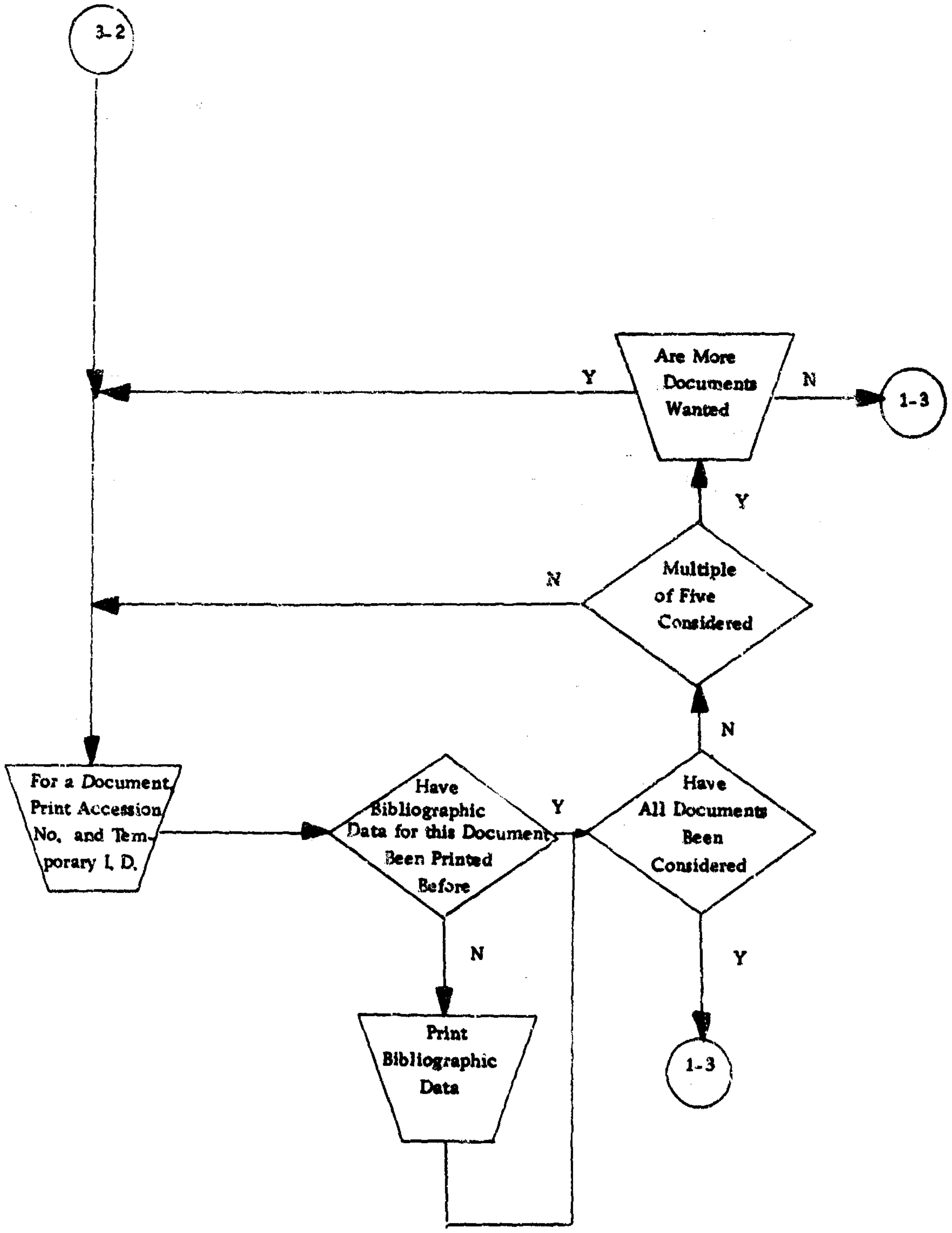
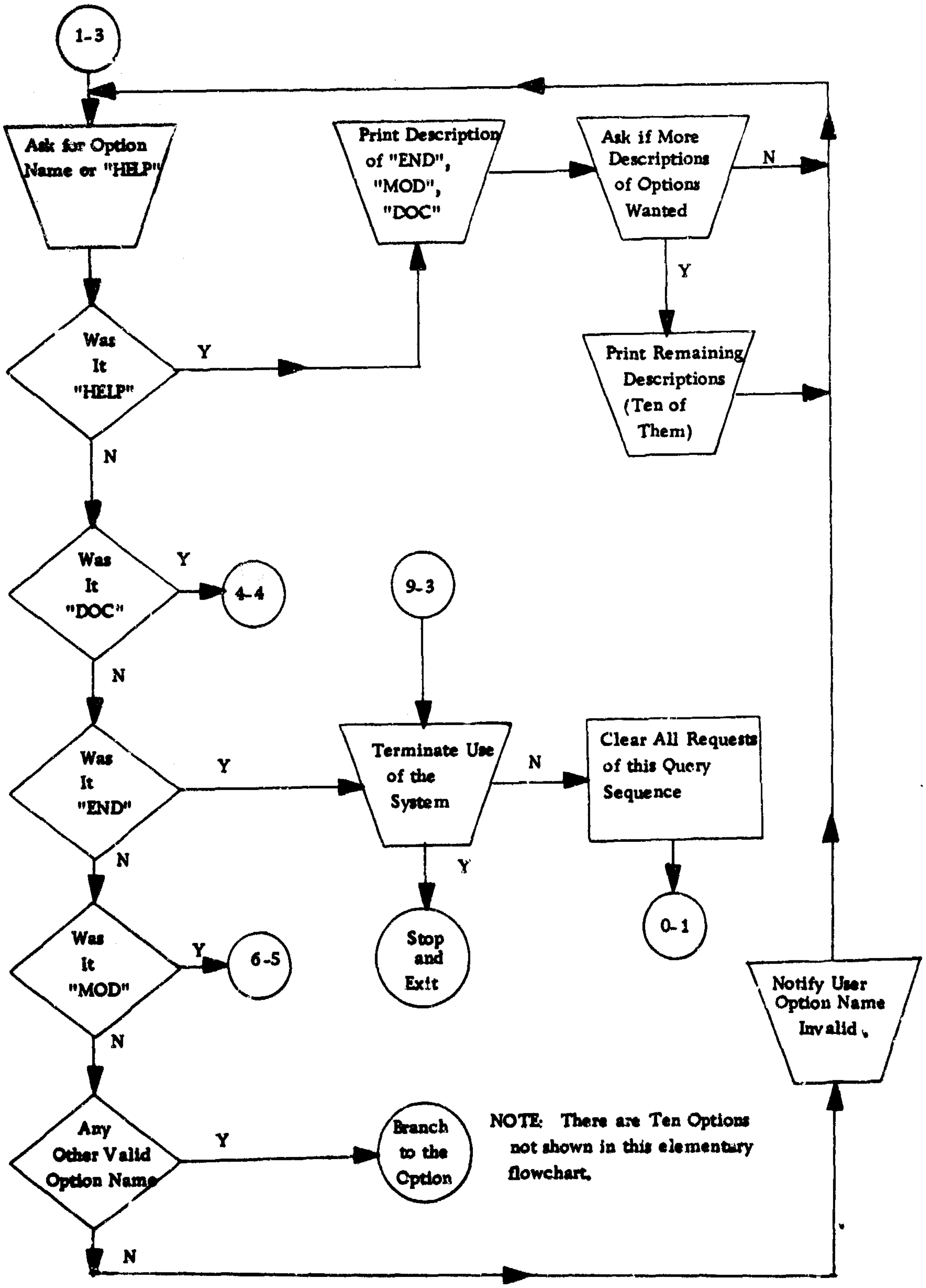

3-2
Are More Documents Wanted
Y
N
1-3
Y
Multiple of Five Considered
N
N
For a Document Print Accession No. and Temporary I. D.
Have Bibliographic Data for this Document Been Printed Before
Y
Have All Documents Been Considered
N
Y
Print Bibliographic Data
1-3

Figure IV-2. TYRO (Cont'd.) TYRO 2


1-3
Ask for Option Name or "HELP"
Was It "HELP"
Y
N
Print Description of "END", "MOD", "DOC"
Ask if More Descriptions of Options Wanted
N
Y
Print Remaining Descriptions (Ten of Them)
Was It "DOC"
Y
4-4
N
9-3
Was It "END"
Y
Terminate Use of the System
N
Clear All Requests of this Query Sequence
N
Y
0-1
Was It "MOD"
Y
6-5
Stop and Exit
Notify User Option Name Invalid.
N
Any Other Valid Option Name
Y
Branch to the Option
N

NOTE: There are Ten Options not shown in this elementary flowchart.

Figure IV-2. TYRO (Cont'd.)

TYRO 3

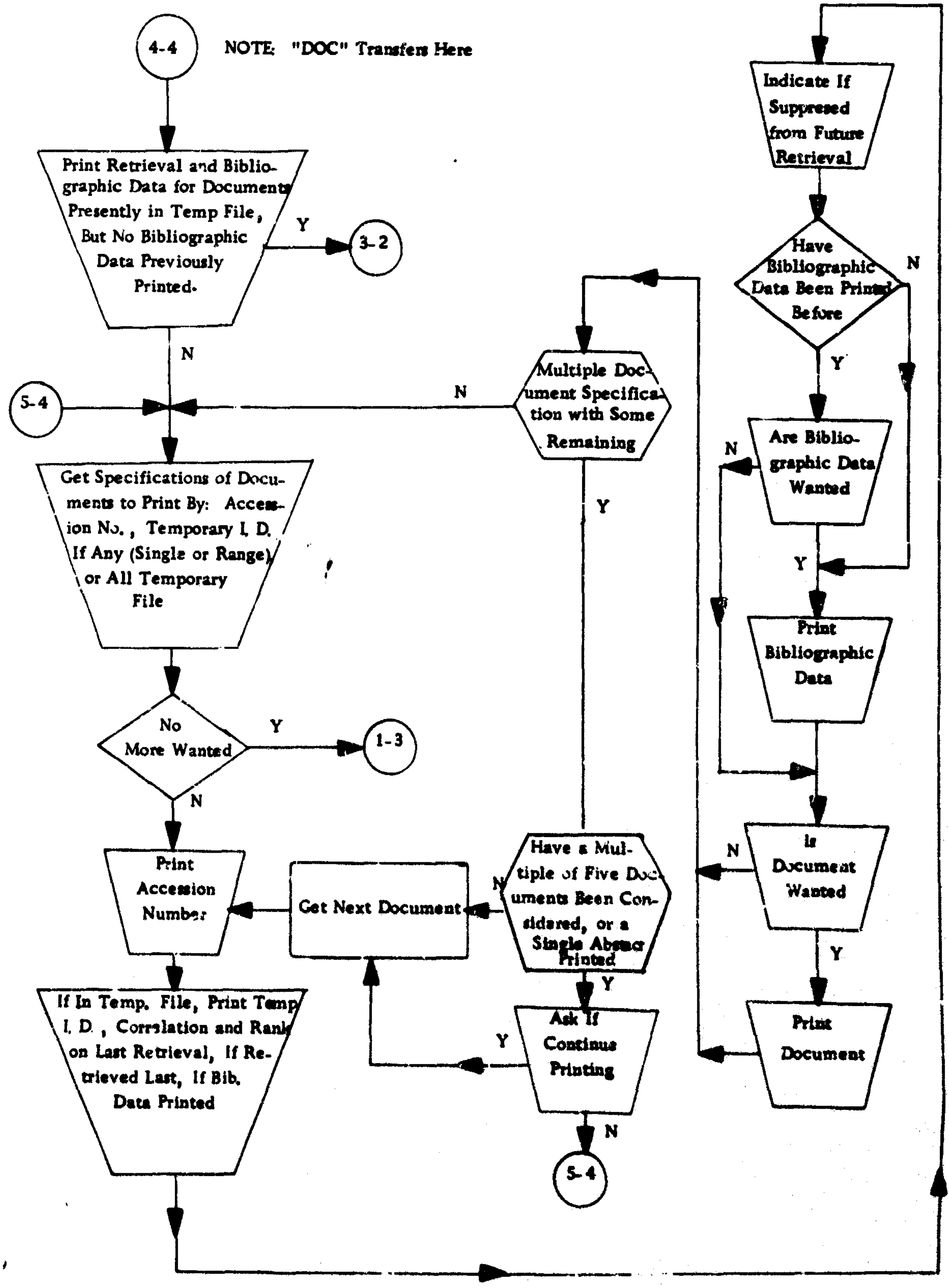
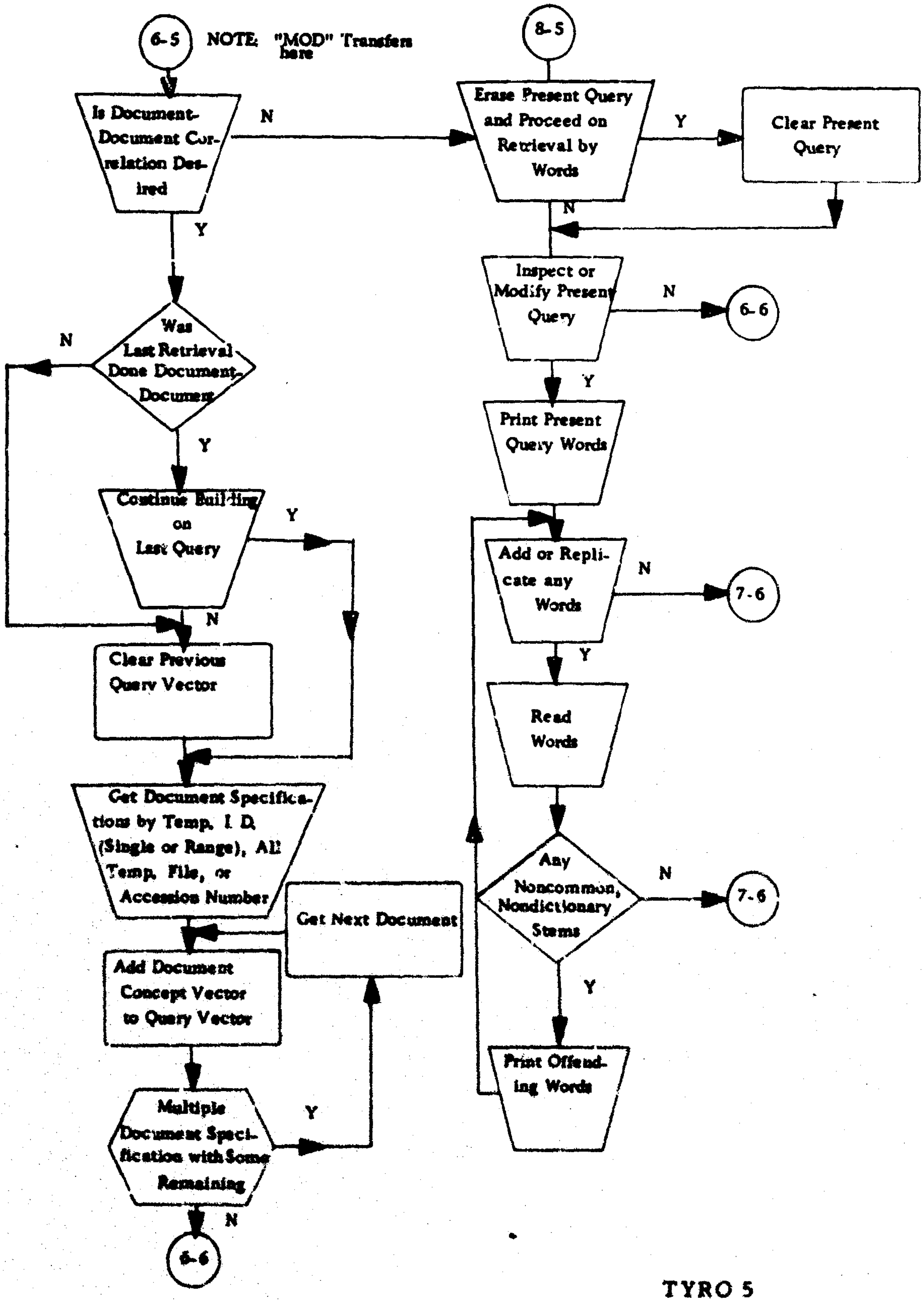
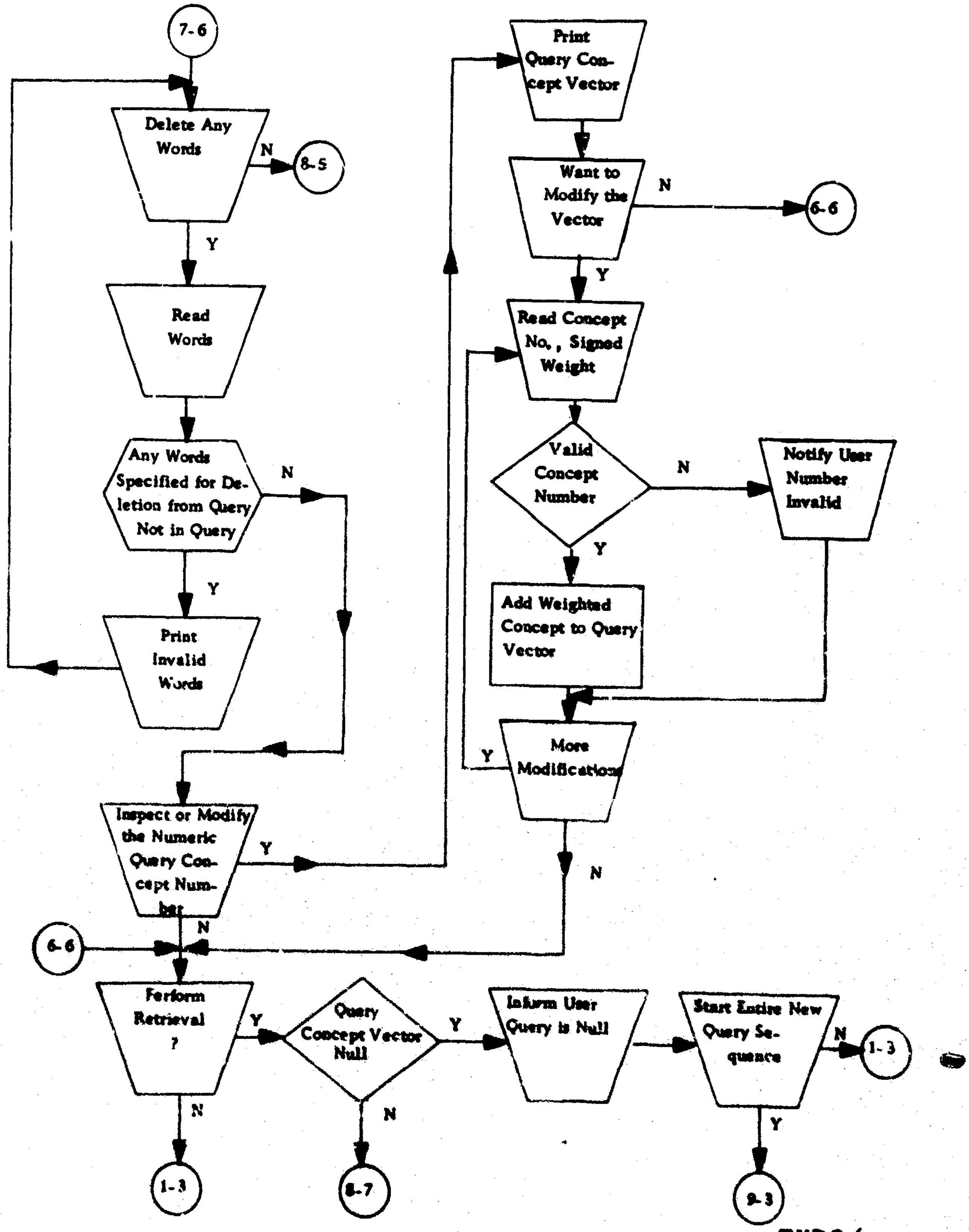

4-4
NOTE: "DOC" Transfers Here
Print Retrieval and Bibliographic Data for Documents Presently in Temp File, But No Bibliographic Data Previously Printed.
Y
3-2
N
5-4
Get Specifications of Documents to Print By: Accession No., Temporary I. D. If Any (Single or Range) or All Temporary File
No More Wanted
Y
1-3
N
Print Accession Number
Get Next Document
If In Temp. File, Print Temp I. D., Correlation and Rank on Last Retrieval, If Retrieved Last, If Bib. Data Printed
N
Multiple Document Specification with Some Remaining
Y
Have a Multiple of Five Documents Been Considered, or a Single Abstract Printed
N
Y
Ask If Continue Printing
Y
N
5-4
Indicate If Suppressed from Future Retrieval
Have Bibliographic Data Been Printed Before
N
Y
Are Bibliographic Data Wanted
N
Y
Print Bibliographic Data
Is Document Wanted
N
Y
Print Document

Figure IV-2. TYRO (Cont'd.) TYRO 4


6-5
NOTE: "MOD" Transfers here
Is Document-Document Correlation Desired
N
Y
8-5
Erase Present Query and Proceed on Retrieval by Words
Y
Clear Present Query
N
Inspect or Modify Present Query
N
6-6
Y
Print Present Query Words
Was Last Retrieval Done Document-Document
N
Y
Continue Building on Last Query
Y
N
Clear Previous Query Vector
Add or Replicate any Words
N
7-6
Y
Read Words
Get Document Specifications by Temp. I. D. (Single or Range), All Temp. File, or Accession Number
Get Next Document
Any Noncommon, Nondictionary Stems
N
7-6
Y
Add Document Concept Vector to Query Vector
Print Offending Words
Multiple Document Specification with Some Remaining
Y
N
6-6

TYRO 5

Figure IV-2. TYRO (Cont'd.)


7-6
Delete Any Words
N
8-5
Y
Read Words
Any Words Specified for Deletion from Query Not in Query
N
Y
Print Invalid Words
Inspect or Modify the Numeric Query Concept Number
Y
N
6-6
Ferform Retrieval ?
Y
N
1-3
Query Concept Vector Null
Y
N
8-7
Inform User Query is Null
Start Entire New Query Sequence
N
1-3
Y
9-3
Print Query Concept Vector
Want to Modify the Vector
N
6-6
Y
Read Concept No., Signed Weight
Valid Concept Number
N
Notify User Number Invalid
Y
Add Weighted Concept to Query Vector
More Modifications
Y
N

TYRO 6

Figure IV-2. TYRO (Cout'd.)

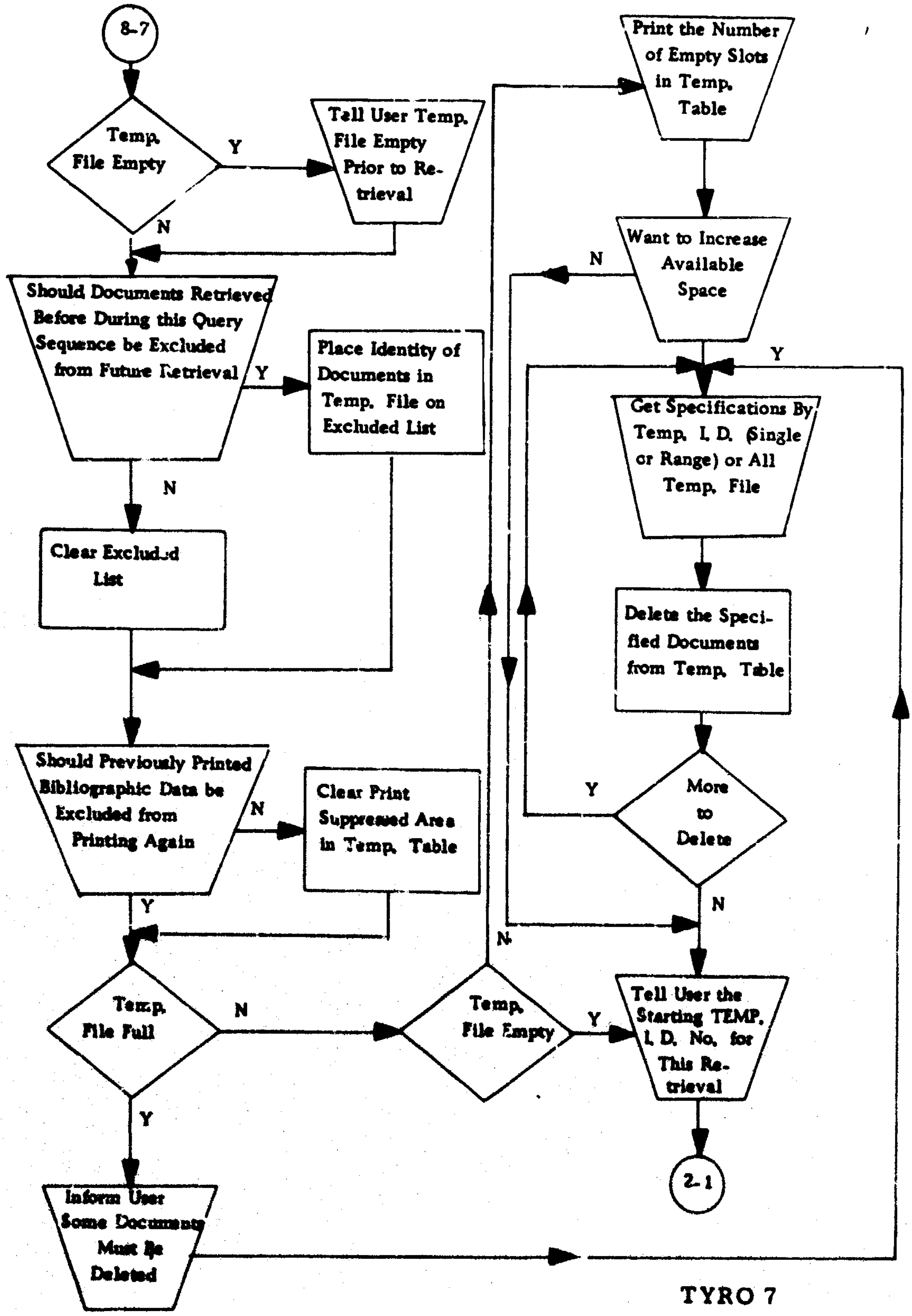

8-7
Temp. File Empty
Y
Tell User Temp. File Empty Prior to Retrieval
N
Should Documents Retrieved Before During this Query Sequence be Excluded from Future Retrieval
Y
Place Identity of Documents in Temp. File on Excluded List
N
Clear Excluded List
Should Previously Printed Bibliographic Data be Excluded from Printing Again
N
Clear Print Suppressed Area in Temp. Table
Y
Temp. File Full
N
Y
Inform User Some Documents Must Be Deleted
Temp. File Empty
N
Y
Print the Number of Empty Slots in Temp. Table
Want to Increase Available Space
N
Y
Get Specifications By Temp. I. D. (Single or Range) or All Temp. File
Delete the Specified Documents from Temp. Table
More to Delete
Y
N
Tell User the Starting TEMP. I. D. No. for This Retrieval
2-1
TYRO 7

Figure IV-2. TYRO (Concluded)

If any words in the initial query are neither common nor found in the dictionary, they are listed for the user's information. If either none of the query words are in the dictionary or the query results in the retrieval of no documents, the user is so informed and asked to enter another query.

When a successful retrieval takes place, the user is told how many documents were retrieved. The accession numbers of the documents are placed in the temporary file. The system then, without questioning the user, starts to print more detailed information about the documents in the temporary file. For each document, the accession number and temporary identification number are printed. Then a check is made to see if bibliographic data for the document have been printed previously during the query sequence-- if not, the bibliographic data are printed. In the former case the output for a document occupies only a single line.

Clearly, users will infrequently want such data printed for the entire set of documents in the temporary file. On the other hand, in order to modify his query intelligently, the user must have some idea of what he has retrieved. After the data for five documents have been printed, the user is asked if more documents are wanted. If they are, five more are printed.

When either all the data for the documents in the temporary file have been printed or the user has decided he has seen enough, he is asked to enter an option name or, in order to get a brief explanation of the options, "HELP". A cry of "HELP" from the user results in the printing of descriptions of MOD, DOC and END options. Now, since it is not the desire to keep the inexperienced user from learning more about the system, he is asked if he wishes to see similar explanations of the remaining ten options, and if he does these are printed. (Similarly, if he attempts to use the option CHG, he is asked if he wishes to see a list of the modes available.)

The user is again asked to enter an option name. Any legitimate option name will be accepted, but this section is concerned with only the basic three. An illegal option name will result in an error message and a request for an option name or "HELP", so that a user who misremembers a name is taken back to the point where aid is available.

The END option, shown on flowchart sheet TYRO 3, causes the user to be asked if he is through with the retrieval system. If he is, the system is shut down; if not, an entire new query sequence is initiated.

The DOC option (TYRO 4) allows the user to obtain more information about the documents presently in the temporary file, or any other documents for which the accession number is known. The user is first asked if he wants only bibliographic information for documents in the present temporary file, with information previously printed suppressed-- just as results after an initial query. If he answers "YES", these data and the temporary file data are made available, with the question "MORE" following every five documents in the bibliographic section. It is expected that this would be done by a user who printed only a small part of the bibliographic data immediately following a retrieval and then wants to obtain more of it.

If the last-mentioned question is answered "NO", the user is asked to specify a document or document set of interest to him. He may do so by entering a single accession number or temporary identification number, or a range of temporary identification numbers, or the word "ALL" to signify all the documents in the temporary file. An illegal entry results in a more detailed explanation of the format required and a request that the user try again.

For each document specified, the accession number is first printed. If the document is in the temporary file, the following are printed: its temporary identification number, rank and correlation on its last retrieval, whether or not the last executed retrieval retrieved the document, and whether or not the bibliographic data for the docum have already been printed.

If the document is suppressed from future retrieval, this fa is stated. Bibliographic data are printed if they have not been printed before; if they have, the operator is asked if they are to be printed aga and the appropriate action is taken. Next the operator is asked if the document itself is to be printed, and prints it in response to an answe f "YES".

If there was only one document specified by accession number or temporary identification, the user is given the opportunity specify more. The process continues as above if he does, or request an option name if he does not.

Printing an entire document may take some time, so even if a set of documents has been specified the user is asked if he wishes to continue after the printing of a document. Similarly, the user is asket if he wishes to continue after the printing of any information from five documents. A negative reply in either case results in a request for an option name, or the specification of other documents to be examined.

The MOD option (TYRO 5) not only allows the user to modify or replace his query, but it also automatically transfers the inexperien d user to sections of other options in order to delete* entries from the temporary file (if desired or required) and perform retrieval**. Upon entrance to MOD, the user is first asked if document-document correl n is to be used as the retrieval method. (Recall that he has started with query words and already retrieved some documents.)

* DEL

** RET

If both document-document correlation is chosen and the last retrieval performed was also based on document-document correlation, the user is given the option of building on the concept vector used in the previous retrieval or starting afresh. He then builds or adds to a query vector by specifying any number of documents by means of single accession numbers, single or ranges of temporary identification numbers, or all the documents in the temporary file. After indicating that no more documents are to be used for the search, the user is asked if he desires to initiate the retrieval.

The point at which the user is asked about starting the retrieval can be reached by another path, which is started when the user rejects document-document correlation. The words forming the last query performed on a query word basis have been retained (with their stems and concept-weight mappings), so the user is given the choice of retaining and building on them or erasing them and building anew set of query words. The system is so designed that a user can inspect, modify and again inspect the set of query words, and so the user is asked if he wishes to inspect or modify the set or not. A negative answer causes the user to be asked if he wishes to initiate retrieval.

If the user indicates that he does wish to inspect or modify the set of query vector words, the present set (with stems and concept-weights) is printed and he is then asked if he wants to add or replicate any words. If he does, he is asked to enter the words. Any noncommon, nondictionary words are reported to the user if they are entered, and he is again given the chance to add or replicate words. The user is then given the opportunity to delete words, and informed if he attempts to delete any words not present and allowed to try again.

Next the user is given the opportunity to inspect the query concept vector directly, and if he so elects it is printed. He may add signed concept number-weight pairs, and is informed of any illegal concept numbers that he attempts to enter.

Use of the above three methods of query vector modification, or some combination of them, eventually leads the user to the point where he is asked if he wants a retrieval performed. It is possible that he wants to return to the point of entering an option name— for example, he might want to have some additional document information printed, and then return to building a document-document correlation query. In such an event, he would answer the question about initiating retrieval in the negative.

When the user indicates that he does want to perform a retrieval, the dialogue processor determines if the query concept vector is null. If it is, the user has obviously become confused, and he is given the opportunity of either starting a new query sequence or resuming the present sequence with a new option name.

Assuming that a retrieval is requested and the query vector is not null, the user is informed if the temporary file is empty. He is asked to specify if documents previously retrieved during the query sequences are to be excluded from re-retrieval or not, and he is asked if printing of bibliographic data already printed once should be allowed or suppressed.

If the temporary file is full, the user is told that he must make space for the documents to be retrieved; if it is partially filled he is given the opportunity to delete documents. Documents to be deleted are specified by accession number, temporary identification number or range of temporary identification numbers. Alternately, the entire temporary file may be deleted.

Then, in order that the user may identify contents of the temporary file with the particular queries retrieving them, he is informed of the starting temporary identification of the documents to be retrieved, and the retrieval is performed.

If no documents are retrieved, the user is so informed and asked to enter an option name or "HELP". If the retrieval is successful, the system continues just as if does after a successful initial retrieval.

IV.4 Additional Options and Special Modes of Operation

In Section IV.3, the essentials of the system required by the inexperienced user are described. The present section is concerned with additional system features, which allow the more experienced user to bring to bear the full power of the system and to operate with total flexibility.

When an option name is requested, any of the following may be entered:

"HELP"	Not truly an option, this aids the user in the selection of the appropriate option name.
"END"	Terminates the present query sequence in order to initiate a new query sequence or sign off.
"MOD"	Modifies or replaces the present query, but continues in the present query sequence.
"DOC"	Prints information concerning documents, both documents retrieved during the present search query and any documents of known accession number.
"OFF"	Prints bibliographic data and documents off-line.
"CHG"	Changes the mode of operation (sequence termination not required)
"CON"	Inspects the concept vectors of documents.
*"RET"	Executes the present retrieval request.
*"DEL"	Deletes unwanted documents retrieved during the present query sequence.
*"SEE"	Inspects the existing query.
*"CLR"	Erases the existing query.
*"WRD"	Adds or deletes query words.

*"DDC" Performs document-document correlation.

*"WGT" Performs direct manipulation of query concept vectors.

(Options marked with "*" are normally called by the system for the inexperienced user.)

Section IV. 5 may be consulted for more information on these options.

In addition to the options selected during a query sequence, a user may set various modes. The inexperienced user will take the default specification in which all modes are deselected, while the more experienced user may select one or more of the following:

1 Select terse dialogue.

2 Skip formation of initial query from words in query sequence.

3 Make available statistical analysis of query.

4 Make available statistical analysis of retrieval.

5 Assume sophisticated user.

Selection of the first mode results in terse, rather than verbose, messages being addressed from the system to the user. Most messages exist in two forms, and the terse form is used by experienced users. The second mode skips initial query formation, and lets the user select an option name immediately after signing onto the system.

If the third mode is selected, the user is asked if he wishes to see the words, stems and concept-weight pairs forming a word-based query. These data are available for printing before the query is executed. Also, he can elect the printing of the query concept vector itself. Although these options also exist under MOD, mode 3 makes them available for analysis of the initial query. Note that this mode does not cause the data to be printed, but simply gives the user the option of printing them.

Similarly, mode 4 is provided so that the user may be asked if he wants the contents of the temporary file (accession number, temporary identificaticn number, rank and correlation when last retrieved, print suppression and whether or not the last executed retrieval retrieved the document) immediately after each retrieval. These data are otherwise available, but mode 4, like mode 3, provides a convenience for the serious student of the system.

Mode 5 simply causes "HELP" to result in the printing of the descriptions of all options, not just the basic three.

IV.5 Detailed Description of the Dialogue Processor

The preceding sections of this chapter are concerned with the general functional characteristics of the dialogue processor. This section is twofold in purpos :, as it presents a detailed structural description of the processor. This is both the final document of the processor's functional appearance and its description from a point of view of design for implementation.

One topic --a set of subroutines required by the dialogue processor --is covered briefly in the subsection immediately following and in greater depth in Section IV.6.

IV.5.1 Three Subroutines - An Overview

Three subroutines are essential to the operation of the dialogue processor. Subroutine OUT prints fixed messages at the remote terminal; YESNO reads the reply to queries which are answered by the user and NUMBER determines the identification of a document or set of documents.

A call to OUT(J) causes the j^{th} standard message to be printed at the remote terminal; a status indicator is returned in j (see

Section VII. 5. 3). Many messages exist in both terse and verbose forms, and if both a terse flag is set (in system common) and a terse form exists, it will be printed rather than the more lengthy form. In flowcharts, symbols like that of Figure IV-3 are used. The call shown, for example,

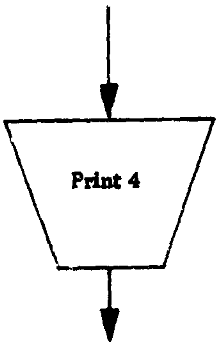


Figure IV-3. Printing a Fixed Message

would cause OUT to print message number four. Section IV. 5. 2 lists the messages; "REFERENCED DOCUMENTS DO NOT EXIST." would be printed unless the terse mode were selected, in which case "INVALID." would be printed.

Subroutine YESNO is used for the many binary decisions required of the user. They must be answered either "YES" or "NO"; this subroutine reads a string from the remote terminal and sets its argument to one if "YES" was read or zero if "NO" was read. The sophisticated user is allowed the word "OPTIONS", which sets the argument to minus one; any other response causes the system to ask the user to 'ANSWER "YES" OR "NO".', and await input.

At several points in the dialogue, the user identifies specific documents. Those documents that have been retrieved during the present query sequence have temporary identification numbers. Any document has, of course, its permanent accession number. Subroutine NUMBER (F1, ARG) is used to return a status flag and accession numbers.

The user specifies documents in the following ways to NUMBER:

- A single temporary identification number (e.g., "13");
- A range of temporary identification numbers (e.g., "10-13");
- The word "ALL" for all documents retrieved during the present query sequence and not subsequently deleted.

F1 must be set to zero prior to the first call on NUMBER; it is an integer. When called, NUMBER returns with F1 set to indicate status as follows:

F1=0	The user has indicated that he wishes to specify no further documents at this time. The contents of ARG are meaningless.
F1=-1	The accession number of a single specified document is contained in ARG. The user may wish to specify more documents, so the calling program should return to NUMBER after processing the document specified in ARG.
F1>0	The user has specified a sequence of documents. The accession number of one of the documents is contained in ARG, and F1 = 1, 2, ... as ARG specifies the first, second, ... document in the sequence. The calling program should continue to return to NUMBER if additional documents in the sequence are required. NUMBER will return with F1 = 0 if the sequence is exhausted. If the calling program is to terminate the sequence early, it can do so providing that the next call (for a new document or document sequence) specifies F1 = 0.

IV.5.2 Fixed Messages to be Printed at the Remote Terminal

Figure IV-4 shows the messages generated by the dialogue processor. Messages one through six are used by YESNO and NUMBER, but are also available to the main routine. Whenever the terse form of a message exists, it is denoted by the letter "T".

1. ANSWER "YES" OR "NO": =

1T. RETRY.

2. ENTER TEMP. ID (SINGLE OR RANGE), ACC. NO. OR "ALL": =

2T. IDENTIFY DOCUMENTS: =

3. MORE?

4. REFERENCED DOCUMENTS DO NOT EXIST.

4T. INVALID.

5. INCORRECT FORMAT. USE FOR EXAMPLE "13" FOR TEMP. ID. NO. 13; "13-33" FOR TEMP. ID. NOS. 13 THRU 33 INCLUSIVE: "A00013" FOR ACCESSION NO. 13; "ALL" FOR ALL DOCUMENTS RETRIEVED IN THIS QUERY SEQUENCE.

5T. FORMAT ERROR.

6. NO DOCUMENTS IN TEMPORARY FILE. THEREFORE, ONLY ACCESSION NUMBERS CAN BE USED TO SPECIFY DOCUMENTS.

6T. TEMP. FILE EMPTY.

7. IS NORMAL OPERATION DESIRED?

8. SKIP INITIAL QUERY?

8T. SKIP INITIAL?

9. ENTER WORDS FOR INITIAL SEARCH QUERY: =

10. THE FOLLOWING ARE NOT USEFUL WORDS FOR RETRIEVAL FROM THIS COLLECTION:

10T. WORDS NOT IN DICTIONARY.

11. NO USEFUL WORDS REMAIN. ANOTHER INITIAL QUERY IS REQUIRED.

11T NO WORDS-RETRIEVAL ABORTED.

12. DO YOU WISH TO CONTINUE IN THE SAME MODE?

12T. SAME MODE?

13. DO YOU WISH TO TERMINATE USE OF THE SYSTEM?

13T. QUIT ?

Figure IV-4. Messages

14. MODE FLAGS ALL OFF. IDENTIFY NUMBERS OF FLAGS TO BE SET ON, OR "0" FOR NORMAL MODE: =

15. DO YOU WANT TO SEE AN ANALYSIS OF YOUR QUERY?

15T. PRINT PRSNT?

16. QUERY WORD STEM WEIGHT ... STEM WEIGHT

17. DO YOU WANT TO SEE THE QUERY CONCEPT VECTOR?

17T. PRINT QUERY VECTOR?

18. STEM WEIGHT STEM WEIGHT ... STEM WEIGHT

19. AT LEAST 50 DOCUMENTS MEET THE SPECIFICATIONS FOR THE PRESENT QUERY.

19T. NO. OF HITS >= 50.

20. THE NUMBER OF DOCUMENTS MEETING YOUR SPECIFICATIONS FOR THE PRESENT QUERY IS

20T. NO. OF HITS =

21. DO YOU WANT A PRINTOUT OF THE TEMPORARY FILE?

21T. PRINT TEMP?

22. ACCESSION NUMBER, TEMPORARY I. D., CORRELATION, BIBLIOGRAPHIC DATA PRINTED BEFORE, RETRIEVED LAST QUERY, RANK ON LAST RETRIEVAL. (Printed as column headings.)

22T. TEMP. ID. CORR. COEF. PRINT BEFORE RET. LAST PREV. RANK (Printed as column headings.)

23. DO YOU WANT BIBLIOGRAPHIC INFORMATION FOR SOME OF THE RETRIEVED DOCUMENTS?

23T. PRINT BIBLIO?

24. PREVIOUSLY PRINTED.

24T. ***

25. BIBLIOGRAPHIC DATA FOR ALL THE TEMPORARY FILE DOCUMENTS HAVE BEEN PRINTED.

25T. THAT'S ALL.

Figure IV-4. Messages (Cont'd.)

26. ON-LINE RETRIEVAL SYSTEM SICNING OFF. (Line feeds, form feed)

27. ENTER NAME OF OPTION (OR "HELP" TO SEE IF THE LIST OF AVAILABLE OPTIONS):=

27T. OPTION:=

28. OPTIONS AVAILABLE ARE:

"END" - TERMINATE THIS SEARCH QUERY SEQUENCE FOR STARTING A NEW SEQUENCE OR SIGNING OFF.

"MOD" - MODIFY OR REPLACE THE PRESENT QUERY AND CONTINUE THE PRESENT QUERY SEQUENCE.

"DOC" - PRINT DATA FOR DOCUMENTS RETRIEVED DURING THIS SEQUENCE OR ANY DOCUMENTS OF KNOWN ACCESSION NUMBER.

28T. "END", "MOD", "DOC".

29. "OFF" - PRINT DOCUMENTS OFFLINE.

"CHG" - CHANGE THE MODE OF OPERATION (QUERY SEQUENCE TERMINATION NOT REQUIRED). A LIST OF MODES IS PROVIDED.

"CON" - INSPECT THE CONCEPT VECTORS OF DOCUMENTS.

*"RET" - EXECUTE THE PRESENT RETRIEVAL REQUEST.

*"DEL" - DELETE UNWANTED DOCUMENTS RETRIEVED DURING THE PRESENT QUERY SEQUENCE.

*"SEE" - INSPECT THE EXISTING QUERY.

*"CLR" - ERASE THE EXISTING QUERY.

*"WRD" - ADD OR DELETE QUERY WORDS.

*"DDC" - PERFORM DOCUMENT-DOCUMENT CORRELATION

*"WGT" - PERFORM DIRECT MANIPULATION OF QUERY CONCEPT VECTORS.

(OPTIONS MARKED WITH "*" ARE NORMALLY CALLED AUTOMATICALLY FOR THE USER BY THE SYSTEM.)

29T. "OFF", "CHG", "CON", "RET", "DEL", "SEE" "CLR", "WRD", "DDC", "WGT".

30. OTHER OPTIONS ALSO ARE AVAILABLE. DO YOU WANT A LIST?

30T. MORE?

Figure IV-4. Messages (Cont'd.)

31. THE OPTION NAME YOU ENTERED DOES NOT EXIST.

31T. INVALID.

32. PRESENT SEARCH QUERY SEQUENCE TERMINATED. A NEW QUERY MAY BE INITIATED AT THIS TIME OR YOU MAY SIGN OFF.

32T. SEQUENCE KILLED.

33. CONTINUE PRINTING FROM THIS SPECIFIED GROUP?

33T. CONTINUE?

34. DO YOU WANT AN EXPLANATION OF THE AVAILABLE MODES?

35. MODES ARE NORMALLY "OFF", AND CAN BE TURNED ON BY TYPING IN A FLAG NUMBER OR SEQUENCE OF FLAG NUMBERS, SUCH AS "1, 3, 5". THE FOLLOWING MODES ARE AVAILABLE:

FLAG NUMBER	ACTION
1	SELECT TERSE DIALOGUE
2	SKIP FORMATION OF INITIAL QUERY IN QUERY SEQUENCE FROM WORDS.
3	MAKE AVAILABLE QUERY WORDS, STEMS, CONCEPTS BEFORE RETRIEVAL
4	MAKE AVAILABLE TEMP TABLE CONTENTS AFTER RETRIEVAL
5	ASSUME ANY OPTION MAY BE USED.

35T. 1 = TERSE, 2 - SKIP INITIAL, 3 = QUERY ANALYSIS, 4 RETRIEVAL ANALYSIS, 5 = ALL OPTIONS.

36. SOME DOCUMENTS ARE TO BE DELETED FROM THE TEMPORARY FILE --

36T. DELETE ACTIVE.

37. BEFORE THE PRESENT RETRIEVAL IS PERFORMED.

37T. BEFORE RETRIEVING.

38. YOU MUST SELECT THE DOCUMENTS TO BE DELETED.

38T. SELECT.

39. ACC. NO. CONCEPT- WEIGHT PAIRS

Figure IV-4. Messages (Cont'd.)

39 T. (null message)

40. (Space over to line up.)

41. TEMPORARY FILE EMPTY PRIOR TO EXECUTION OF PRESENT QUERY.

41T. TEMP. FILE EMPTY TO START.

42. DOCUMENTS RESULTING FROM THIS RETRIEVAL WILL HAVE TEMP. NOS. STARTING WITH

42T. TEMP. ID. STARTS WITH

43. SHOULD DOCUMENTS RETRIEVED PREVIOUSLY DURING THIS QUERY SEQUENCE BE EXCLUDED FROM RE-RETRIEVAL?

43T. EXCLUDE PREVIOUS?

44. SHOULD PRINTING OF BIBLIOGRAPHIC DATA PREVIOUSLY PRINTED BE SUPPRESSED?

44T. SUPPRESS PREV. PRINTED?

45. SPACES EXIST IN TEMPORARY FILE FOR NEW RETRIEVALS. MORE SPACE DESIRED?

45T. ENTRIES OPEN. MORE?

46. TABLE OF DOCUMENTS RETRIEVED DURING THIS QUERY SEQUENCE IS FULL. SPACE MUST BE MADE BEFORE EXECUTING ANOTHER QUERY.

46T. TABLE FULL.

47. PRINT ONLY RANKING AND BIBLIOGRAPHIC DATA (NO ABSTRACTS) FOR DOCUMENTS RETRIEVED DURING THIS QUERY SEQUENCE, EXCLUDING BIBLIOGRAPHIC DATA ALREADY PRINTED?

47T. TEMP. DOCS. ONLY, SHORT FORM?

48. THIS DOCUMENT EXCLUDED FROM RE-RETRIEVAL DURING PRESENT QUERY SEQUEN

48T. ON NO-NO LIST.

49. BIBLIOGRAPHIC DATA PRINTED BEFORE IN THIS QUERY SEQUENCE. PRINT AGAIN?

49T. PRINT BIBLIOGRAPHIC AGAIN?

50. PRINT ABSTRACT?

51 DO YOU WANT TO ERASE THE PRESENT QUERY AND DO DOCUMENT-DOCUMENT SEARCHING?

51T. DOC.-DOC.?

Figure IV-4. Messages (Cont'd.)

52. NOW SPECIFY THE DOCUMENTS FOR CORRELATION.

52T. (null message).

53. DO YOU WANT TO SEE OR MODIFY THE WORDS FORMING THE QUERY?

53T. QUERY WORD ACTION?

54. THE PRESENT QUERY WORDS ARE:

54T. PRESENT:

55. DO YOU WANT TO ADD OR REPLICATE ANY WORDS?

55T. ADD WORDS?

56. ENTER WORDS:=

57. DO YOU WANT TO DELETE ANY WORDS?

57T. DELETE?

58. YOU CANNOT DELETE A WORD THAT IS NOT ALREADY IN THE QUERY:

58T. NOT IN QUERY:

59. DO YOU WANT TO BUILD ON THE PREVIOUS DOCUMENT-DOCUMENT SEARCH?

59T. CONTINUE PREV. ?

60. DO YOU WANT TO INSPECT OR DIRECTLY MODIFY THE QUERY CONCEPT VECTOR?

60T. DIRECT CON. VECT. ACTION?

61. DO YOU WANT TO MODIFY THIS VECTOR?

61T. MODIFY?

62. ENTER CONCEPT, WEIGHT PAIR (E. G. "1203, -0.125"):

62T. ENTER PAIR:=

63. ENTER PAIR=

64. INVALID CONCEPT NUMBER. DO YOU WANT TO TRY ANY

64T. INVALID.

65. DO YOU WANT A RETRIEVAL PERFORMED WITH THE PRESENT QUERY VECTOR?

Figure IV-4. Messages (Cont'd.)

65T. RETRIEVE?

66. A RETRIEVAL CANNOT BE PERFORMED BECAUSE YOUR PRESENT QUERY VECTOR IS NULL. DO YOU WANT TO START A NEW QUERY SEQUENCE?

66T. NULL VECTOR WANT NEW SEQUENCE?

67. READY TO PRINT DOCUMENTS OFF LINE.

67T. OFF LINE PRINT.

68. SPECIFY FIRST DOCUMENT OR DOCUMENT GROUP TO PRINT.

68T. (null message)

69. DO YOU WANT TO PERFORM MORE DOCUMENT-DOCUMENT SEARCHING?

69T. MORE DOC-DOC?

70. DO YOU WANT TO ERASE COMPLETELY YOUR PRESENT QUERY AND ENTER NEW QUERY WORDS?

70T. TOTALLY REPLACE PRESENT QUERY?

71. THE PRESENT QUERY CONCEPT VECTOR IS:

71T. PRESENT QUERY:

72. THE PAST QUERIES AND QUERY CONCEPT VECTORS HAVE BEEN CLEARED.

72T. CLEARED.

73. ILLEGAL SELECTION ; REQUEST IGNORED.

Figure IV-4. Messages (Concluded)

IV.5.3 Major Entry Points

The initial entry point, and the entry points for the different options are located on flowchart sheets as indicated below:

Name	Sheet
Initial	1
CHG	13
CLR	7
CON	14
DDC	6
DEL	11
DOC	12
END	11
MOD	6
OFF	13
RET	10
SEE	7
WGT	9
WRD	7

IV.5.4 Arrays of Major Importance

The following arrays are referred to by name in the flowcharts.

Name	Contents
NONO	Accession numbers of documents excluded from future retrieval.
TEMP	Temporary file: accession number, temporary identification number, correlation coefficient and rank when last retrieved, print suppression flag and flag indicating if retrieved on last executed retrieval.

Name	Content
PRESNT	Words for queries: the words, their stems and concept-weight pair mappings.
QUERY	The query concept vector.

IV. 5.5 Variables Global to DIALOGUE

The following variable names are consistently used for major linkages in the dialogue processor.

Name	Type	Use
IX	I	Next available temporary identification number.
JX	I	Number of entries presently in TEMP.
NEWQ	L	Mode setting precedes initial query.
RFLG	L	Present query not initial.
DEFLG	L	DEL entered through RET.
WFLG	L	MOD has altered PRESNT.
SEEFLG	L	SEE activated by MOD.
WRDFLG	L	WRD activated by MOD.
DOCDOC	L	Last retrieval in present sequence used document-document correlation.
TERSE	L	Terse dialogue: Mode 1 selected.
SKIP1	L	Skip initial query: Mode 2 selected.
PRINTQ	L	QUERY available immediately before retrieval: Mode 3 selected.
PRINTR	L	TEMP available immediately after retrieval: Mode 4 selected.
OPTION	L	HELP prints all options: Mode 5 selected.

IV.5.6 Detailed Flowcharts for the Dialogue Processor

Figure IV-5 is the detailed flowchart for the dialogue processor.

IV.6 Three Subroutines - Detailed Description

This section gives a more detailed description of the subroutines OUT, YESNO and NUMBER than is presented in Section IV.5.1, where they are first introduced.

IV.6.1 OUT(J)

A table of numbered messages exists, some of the messages being available in both "normal" and "terse" forms. A mode flag selects the form of a message to be printed.

A call to OUT(J) causes the normal form of message number J to be printed at the remote Teletype if the terse flag is off; the terse form is used if the flag is on. Labelled common is used by mode flag storage. If there is no message J, then an error message containing J for identification of the erroneous call is printed. This would result from a programming error.

See Section VII.5.3 for a flowchart of OUT.

IV.6.2 YESNO(I)

Many system-generated queries must be answered either "yes" or "no"; this subroutine reads a string from the remote terminal and sets its arguments to one if "yes" was read or zero if "no" was read. The sophisticated user is allowed the word "options", which sets the argument to minus one; any other response causes the system to ask the user to 'ANSWER "YES" OR "NO".', and repeats the query.

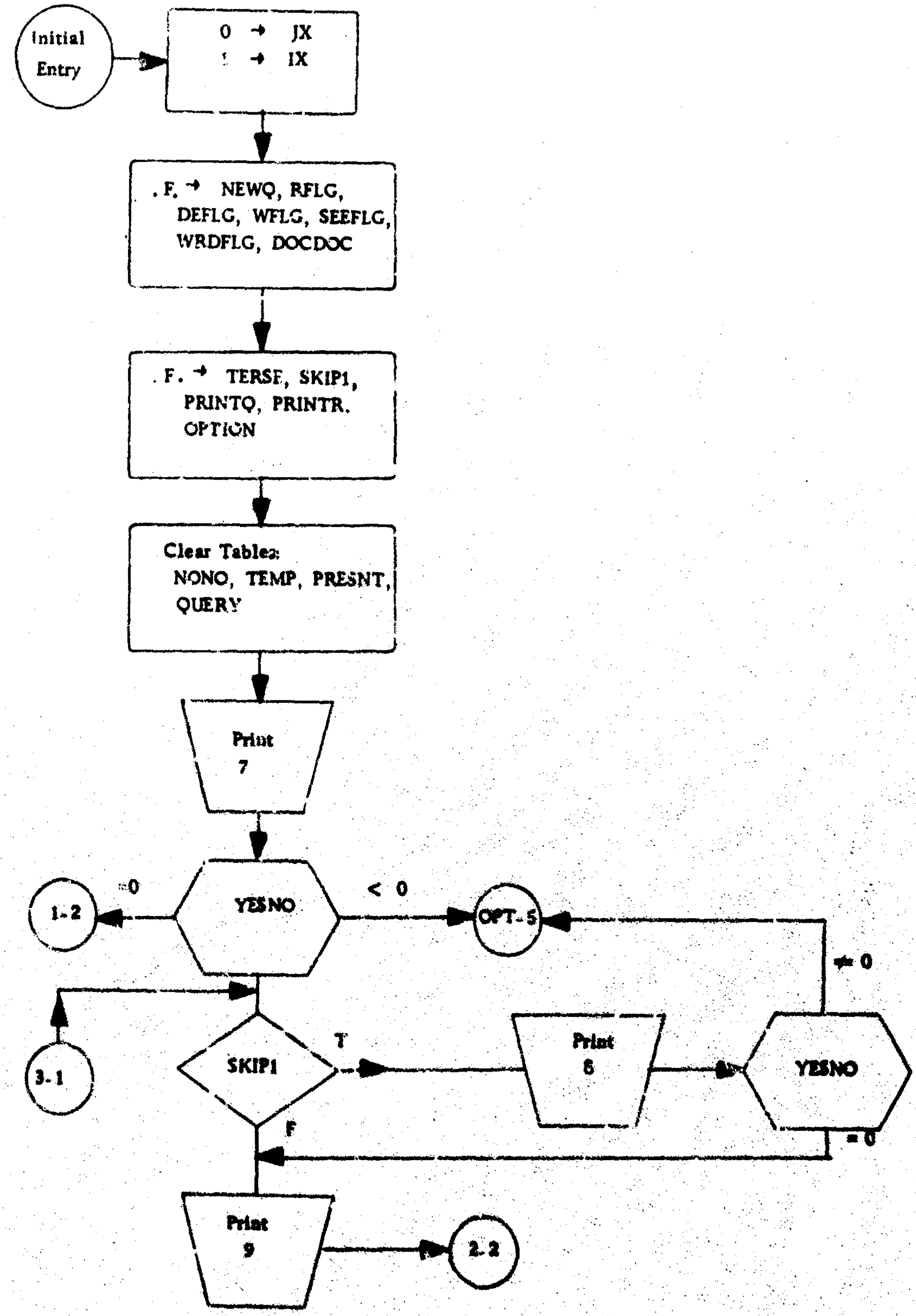

Initial Entry
0 → JX
1 → IX
. F. → NEWQ, RFLG, DEFLG, WFLG, SEEFLG, WRDFLG, DOCDOC
. F. → TERSE, SKIP1, PRINTQ, PRINTR. OPTION
Clear Tables: NONO, TEMP, PRESNT, QUERY
Print 7
=0
1-2
YESNO
< 0
OPT-5
≠ 0
3-1
SKIP1
T
Print 8
YESNO
F
= 0
Print 9
2.2

Figure IV-5. DIALOGUE 1

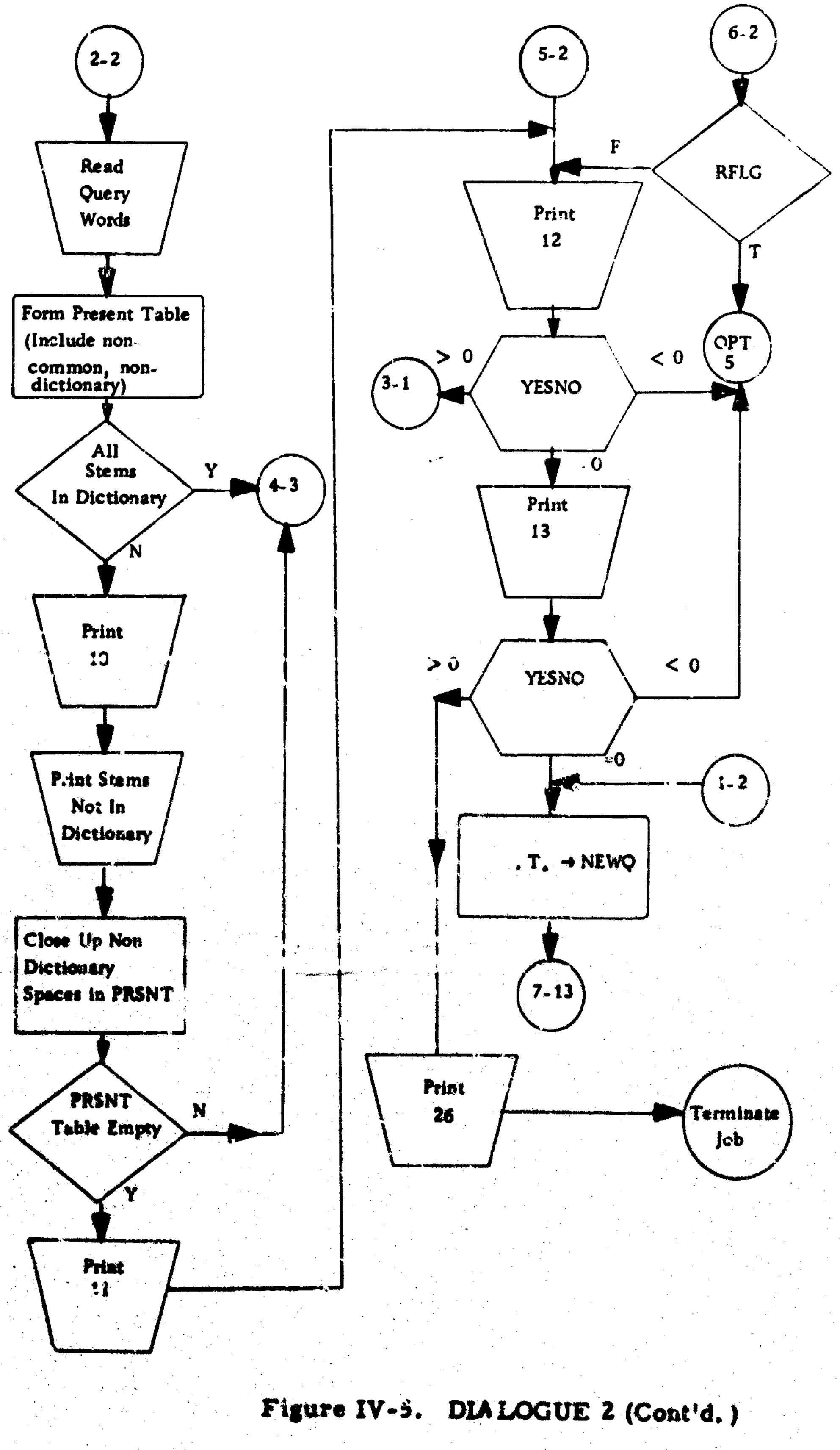


Figure IV-5. DIALOGUE 2 (Cont'd.)

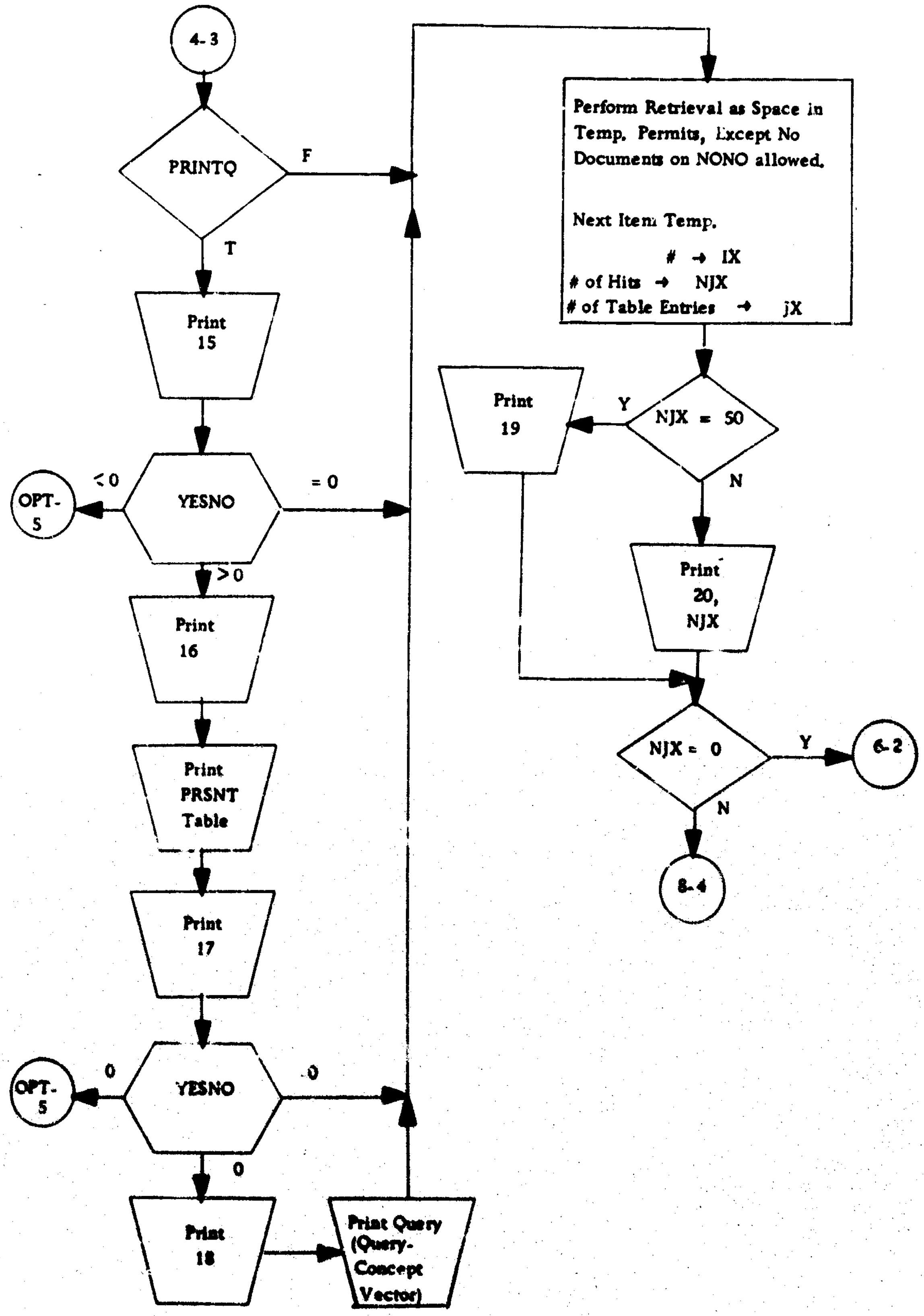


Figure IV-5. DIALOGUE 3 (Cont'd.)

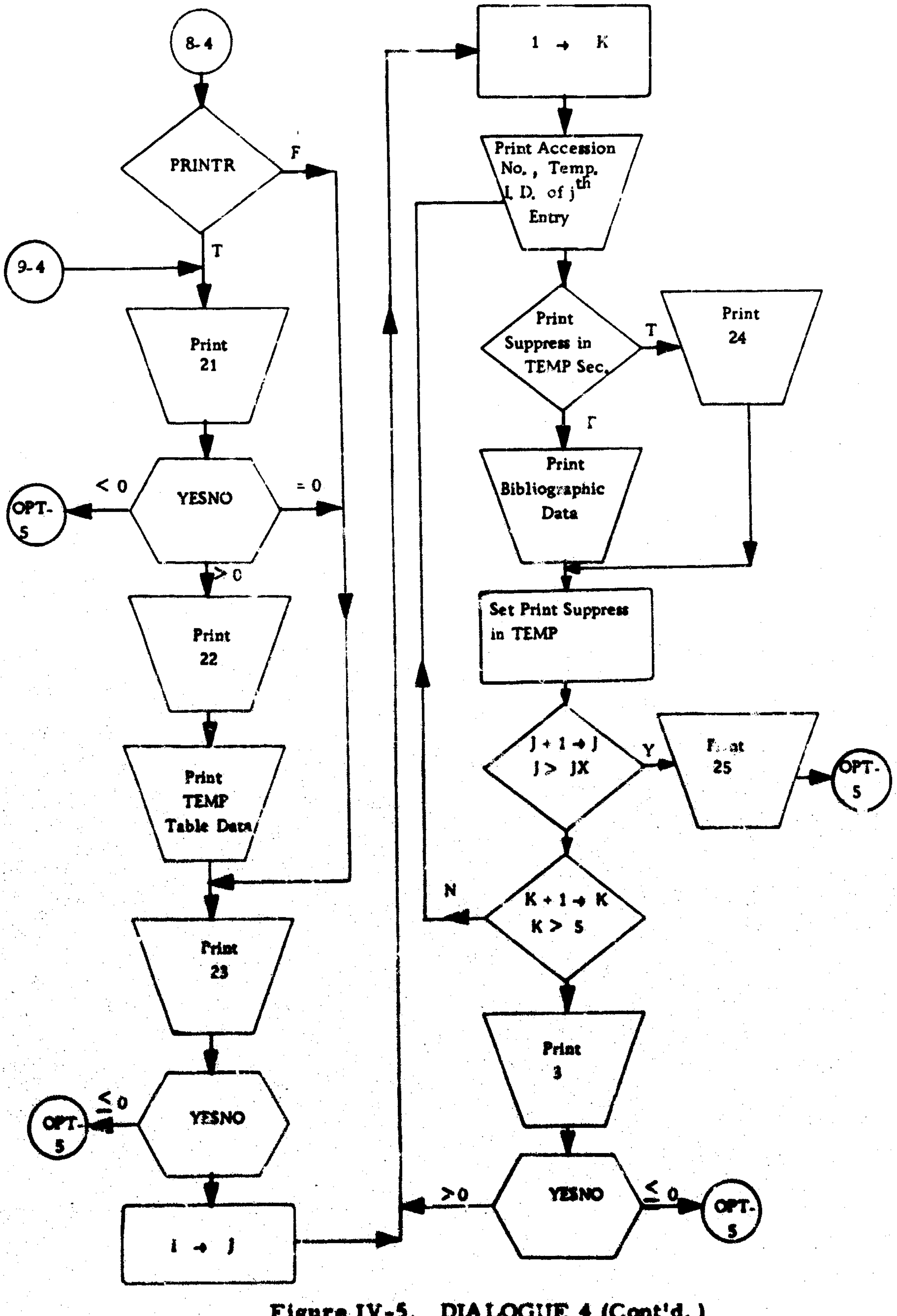


Figure IV-5. DIALOGUE 4 (Cont'd.)

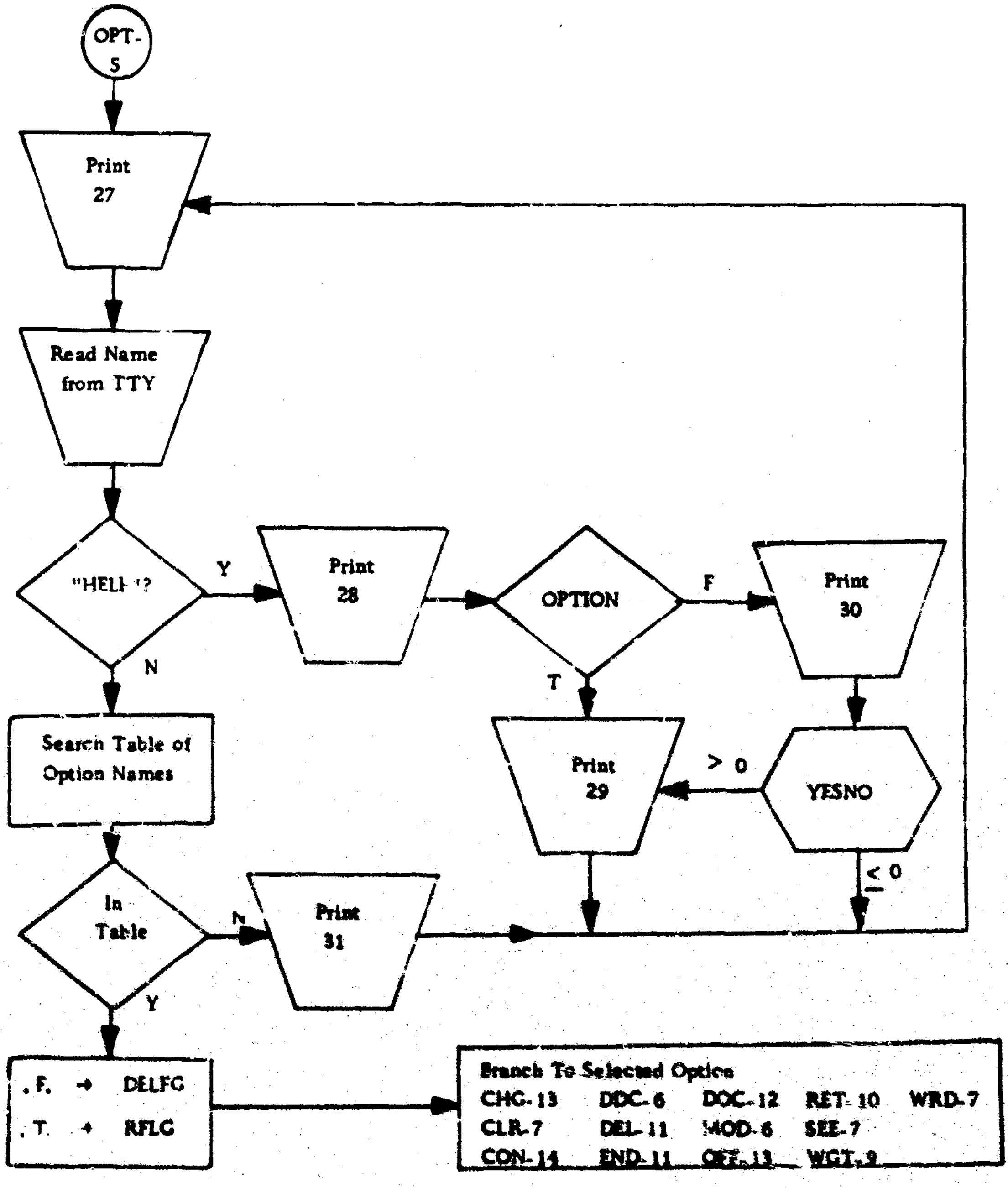


Figure IV-5. DIALOGUE 5 (Cont'd.)

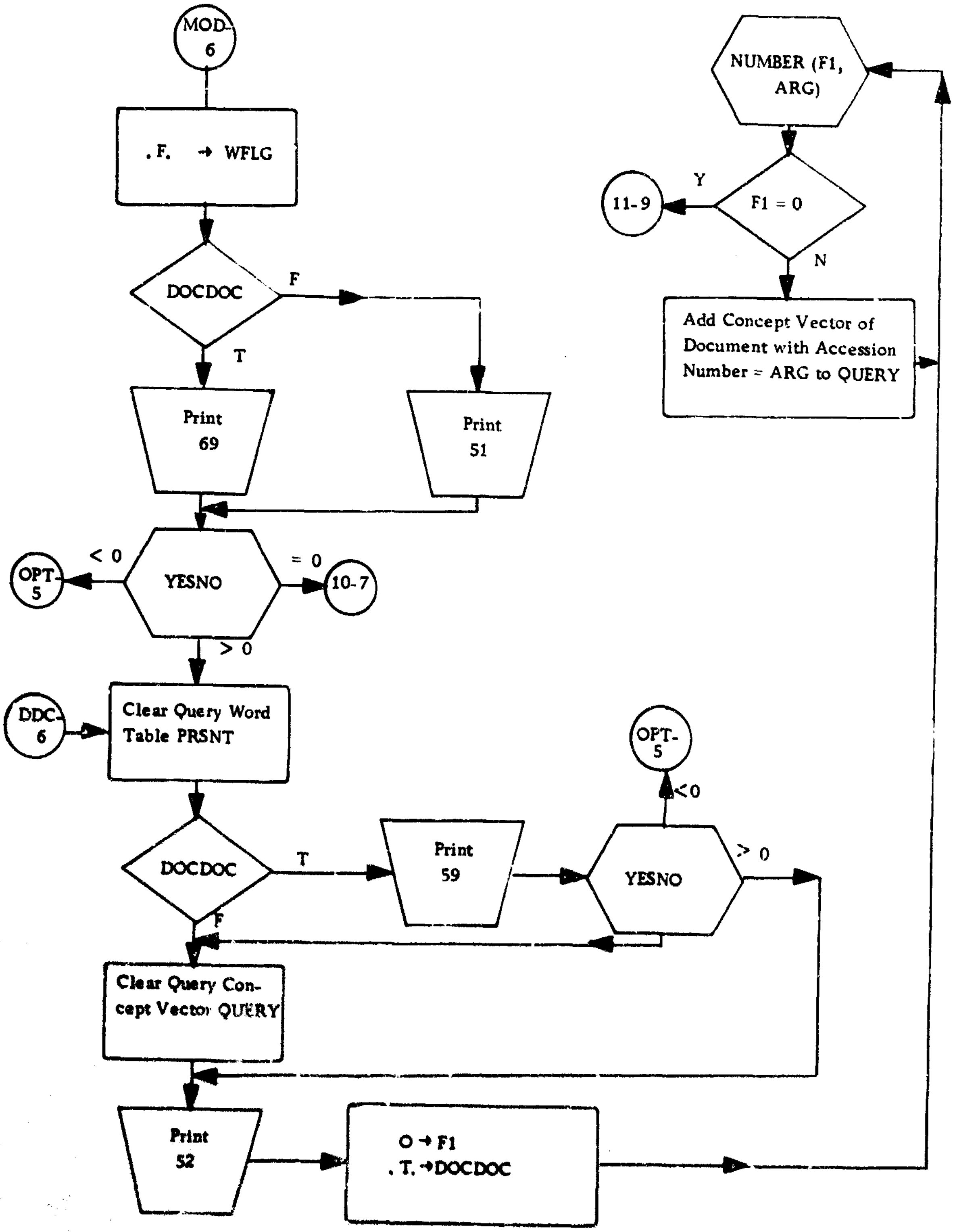


Figure IV-5. DIALOGUE 6 (Cont'd.)

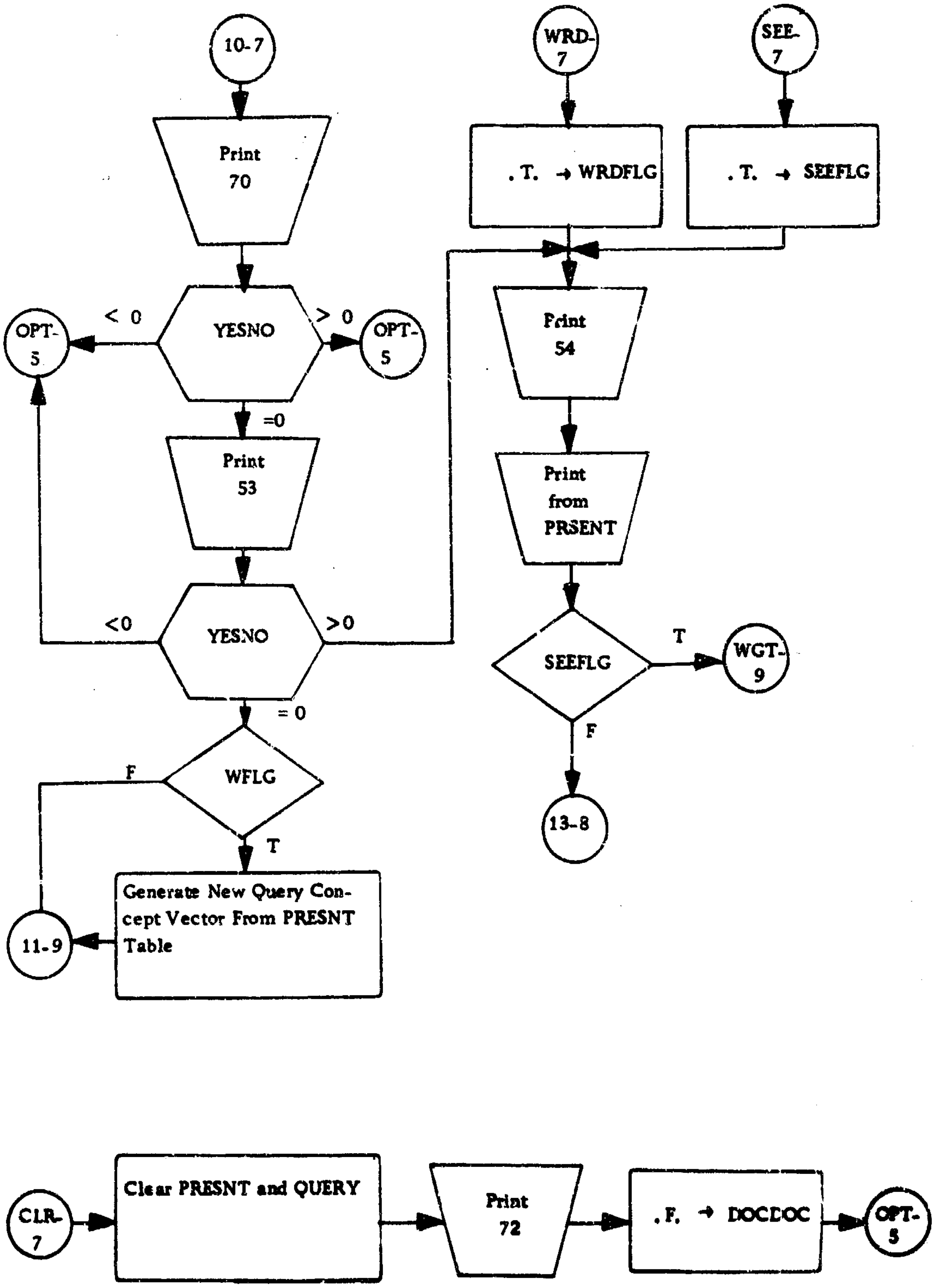


Figure VI-5. DIALOGUE 7 (Cont'd.)

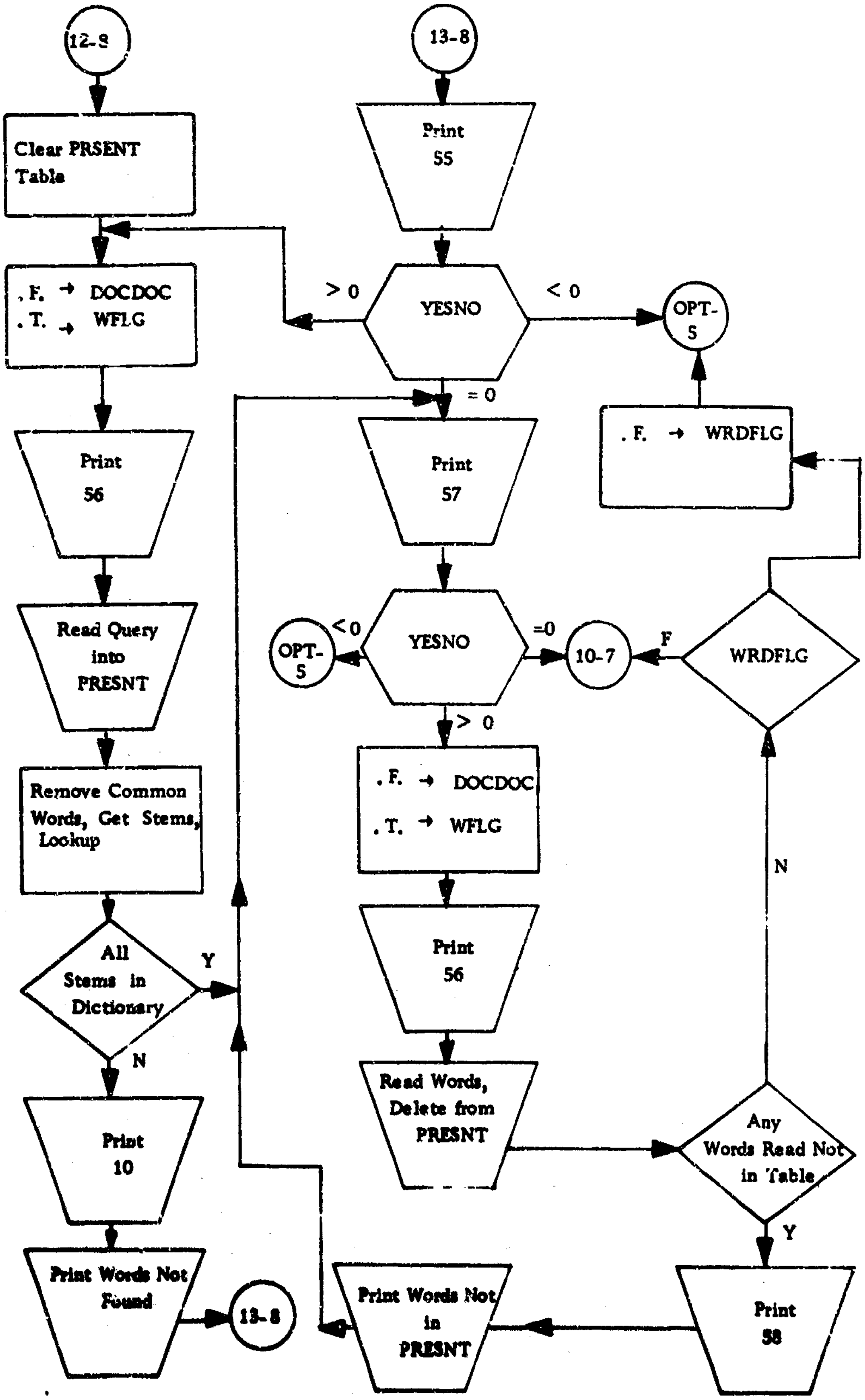


Figure IV-5. DIALOGUE 8 (Cont'd.)

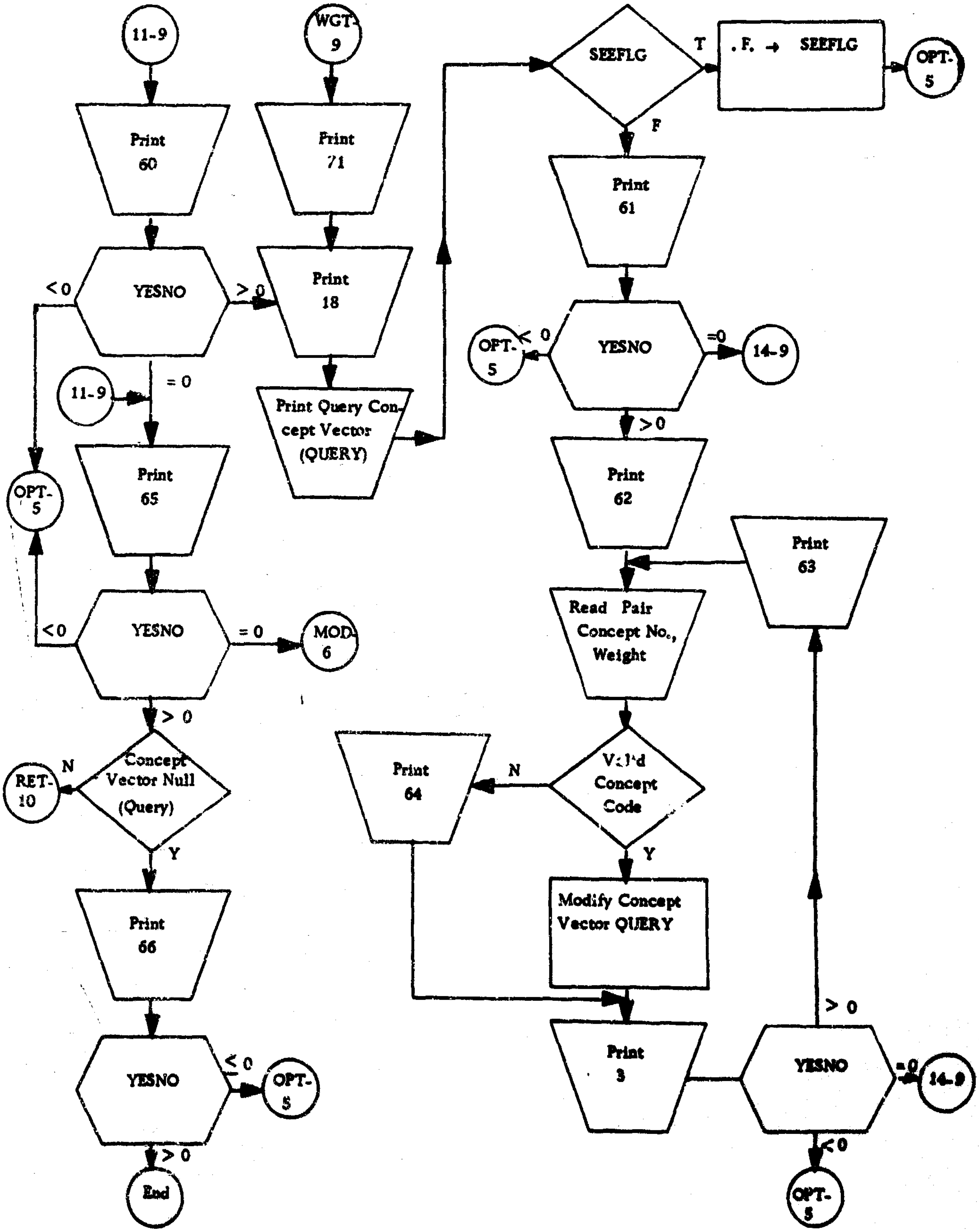


Figure IV-5. DIALOGUE 9 (Cont'd.)

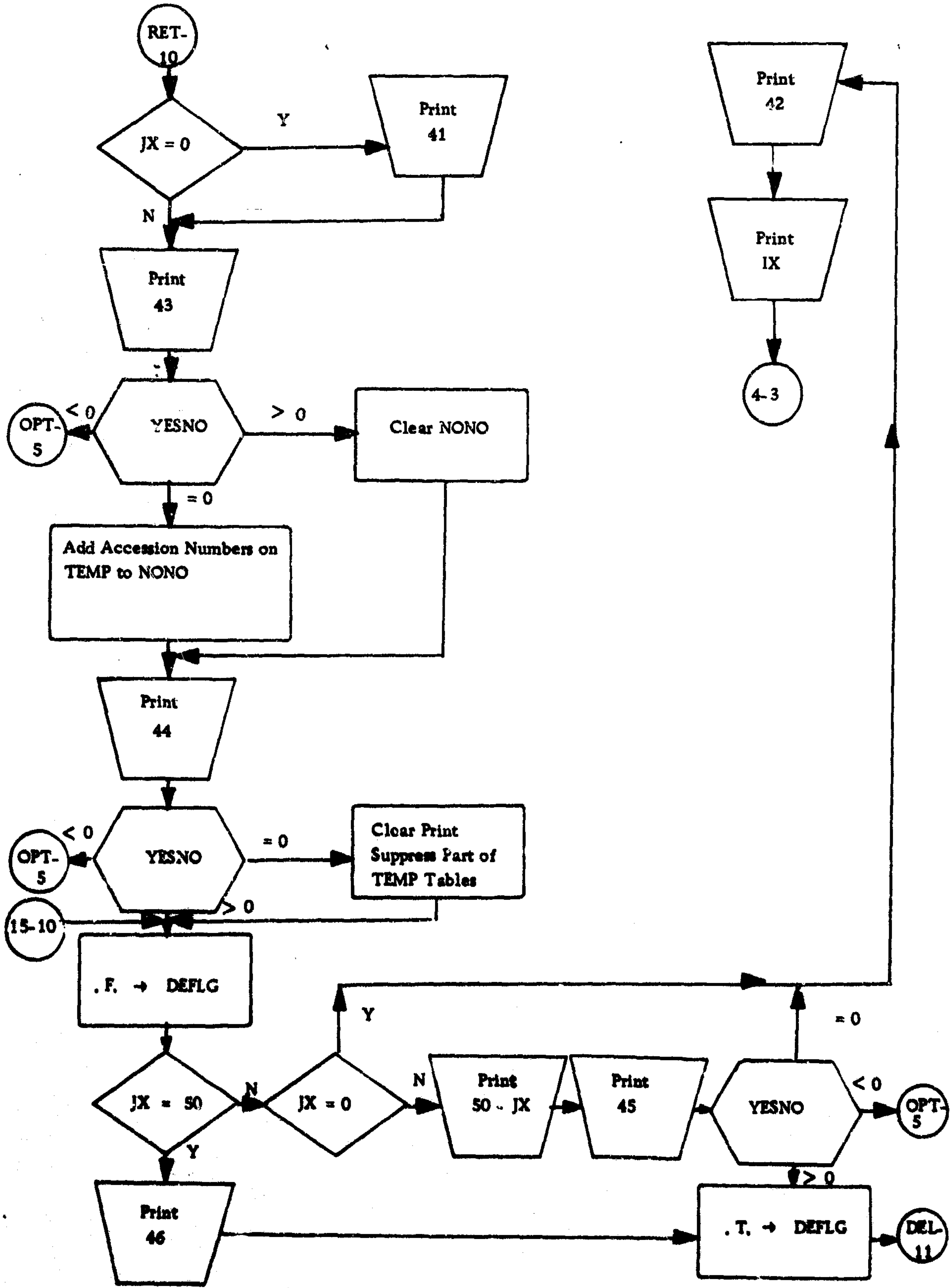


Figure IV-5 . DIALOGUE 10 (Cont'd.)

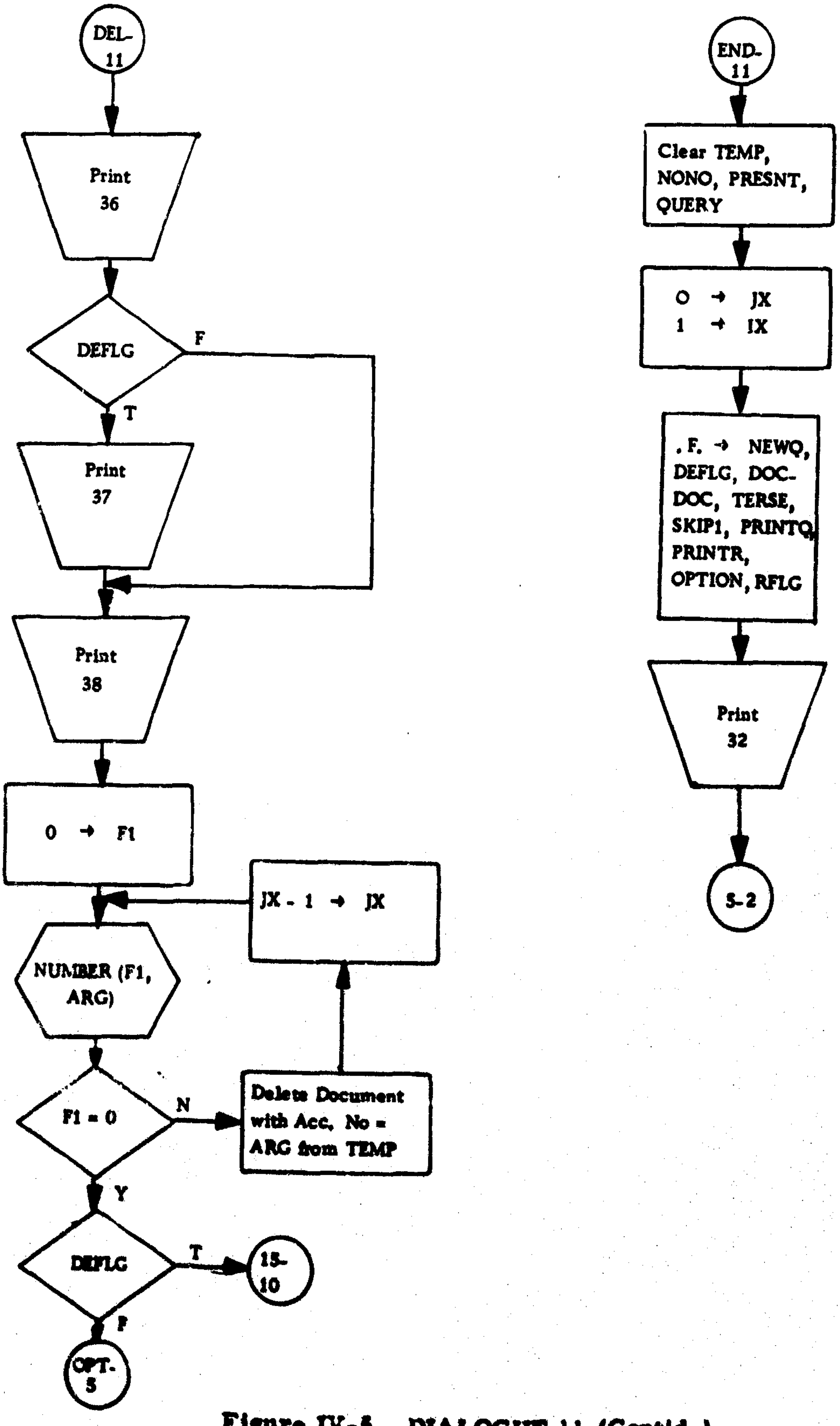

DEL-11
Print 36
DEFLG
F
T
Print 37
Print 38
0 → F1
JX - 1 → JX
NUMBER (F1, ARG)
F1 = 0
N
Delete Document with Acc. No = ARG from TEMP
Y
DEFLG
T
15-10
F
OPT-5
END-11
Clear TEMP, NONO, PRESNT, QUERY
0 → JX
1 → IX
.F. → NEWQ, DEFLG, DOC-DOC, TERSE, SKIP1, PRINTQ, PRINTR, OPTION, RFLG
Print 32
5-2

Figure IV-5. DIALOGUE 11 (Cont'd.)

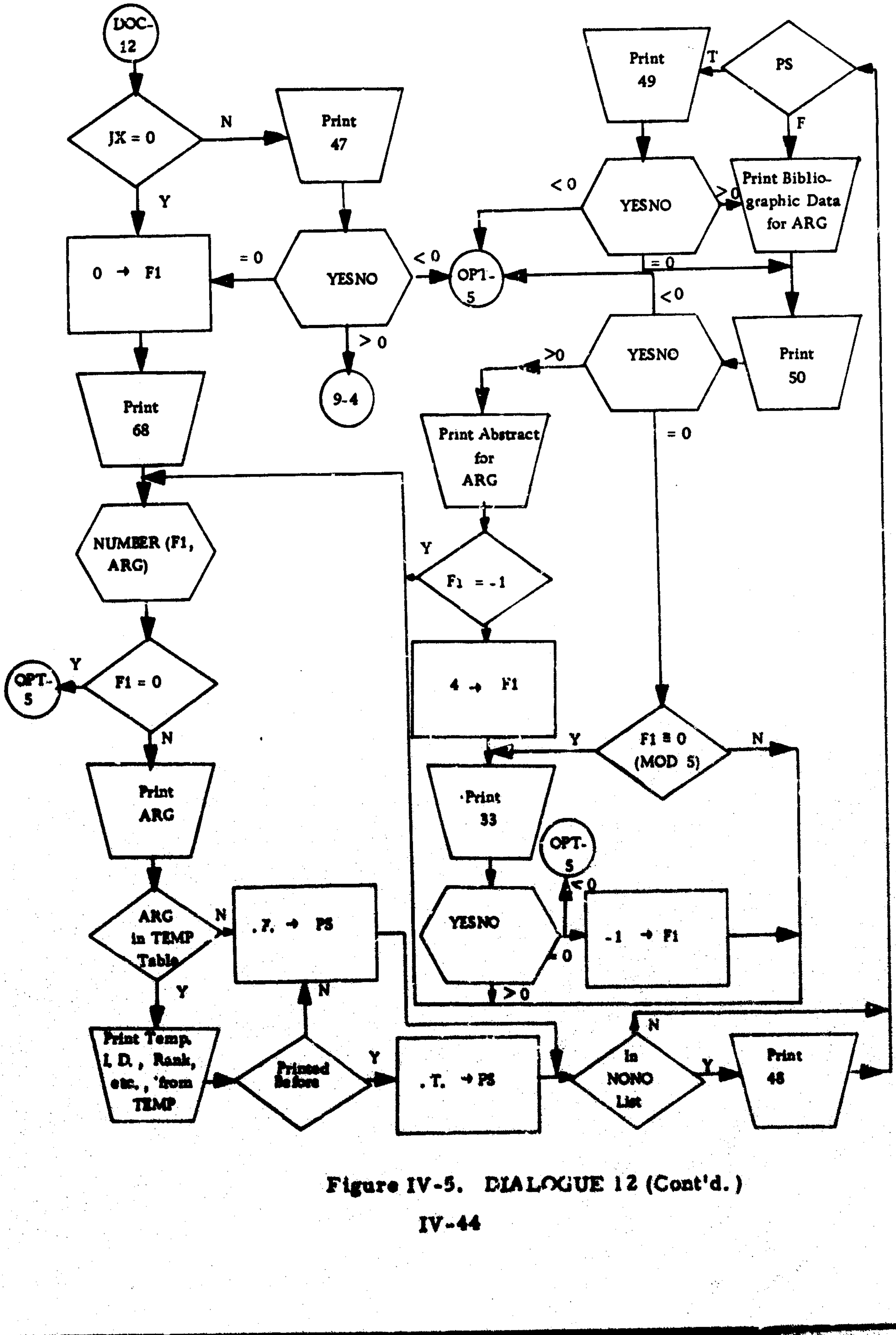


Figure IV-5. DIALOGUE 12 (Cont'd.)

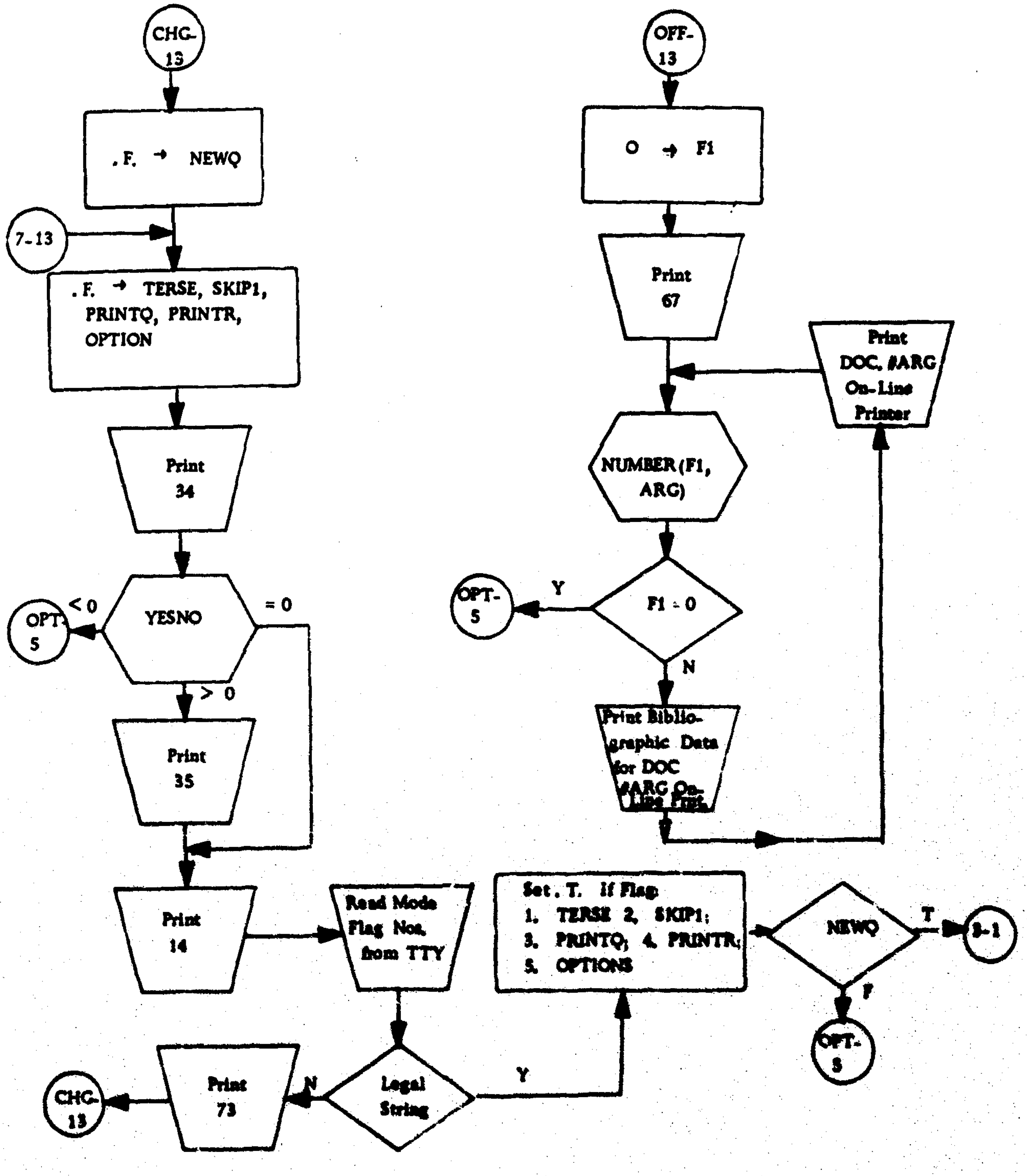


Figure IV-5. DIALOGUE 13 (Cont'd.)

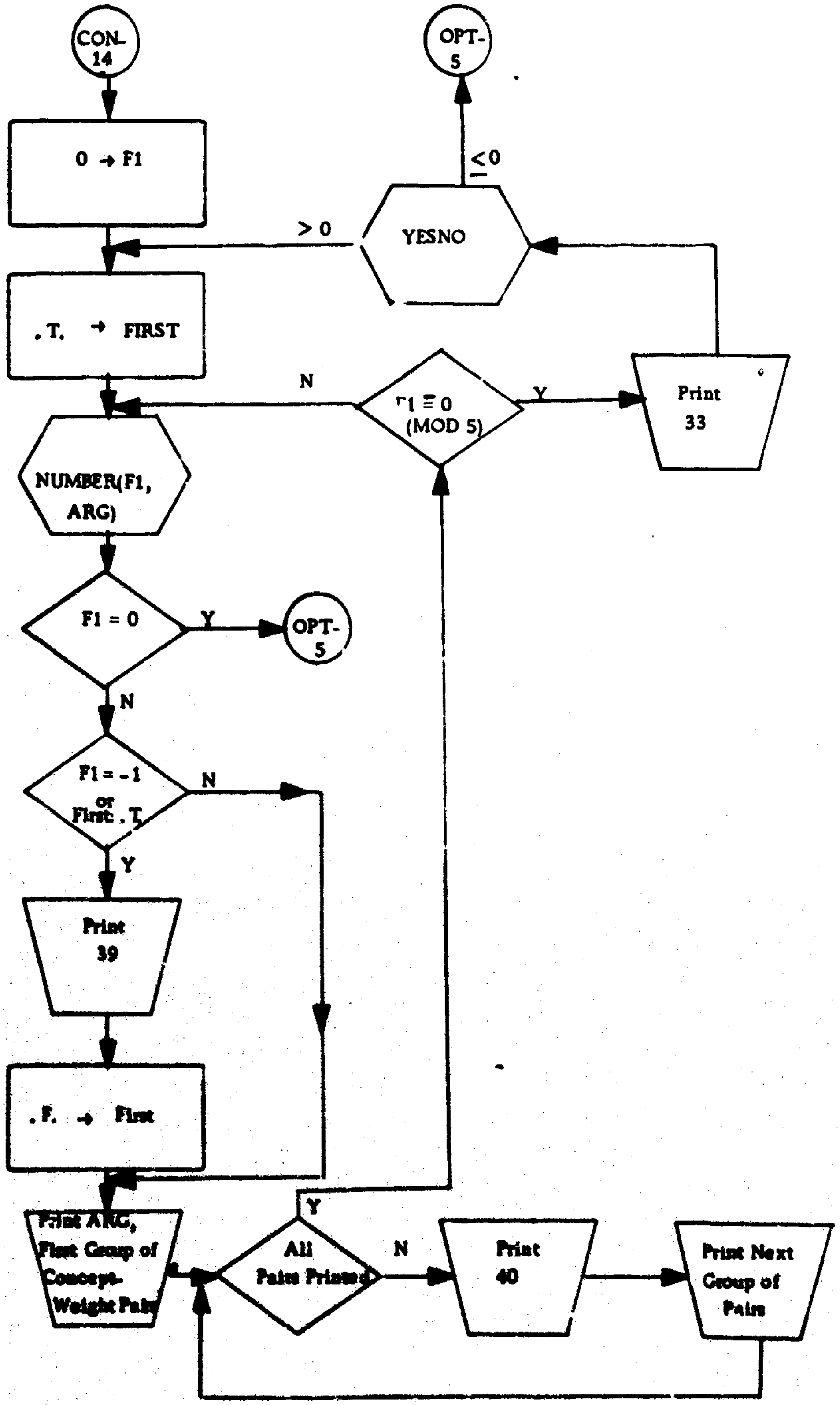


Figure IV-5. DIALOGUE 14 (Concluded)

In many applications it will be useful to code such a routine as an arithmetic function, so that the statement

```
IF (YESNO(I)) 1,2,3
```

branches to 1 for "options", to 2 for "no" and to 3 for "yes". Figure IV-6 shows the flowchart.

IV.6.3 NUMBER (F1,ARG)

At several points in the dialogue, the user identifies specific documents. If these documents have been retrieved during the present query sequence, they have temporary identification numbers. Any document has, of course, its permanent accession number.

The user specifies documents in the following ways to NUMBER:

- A single temporary identification number (e.g., "13");
- A range of temporary identification numbers (e.g., "10-13");
- A single accession number preceded by the letter "A" (e.g., "A09871");
- The word "ALL" for all documents retrieved during the present query sequence and not subsequently deleted.

F1 must be set equal to zero prior to the first call on the NUMBER; it is an integer. When called, NUMBER returns with F1 set to indicate status as follows:

F1 = 0. The user has indicated that he wishes to specify no further documents at this time. The contents of ARG are meaningless.

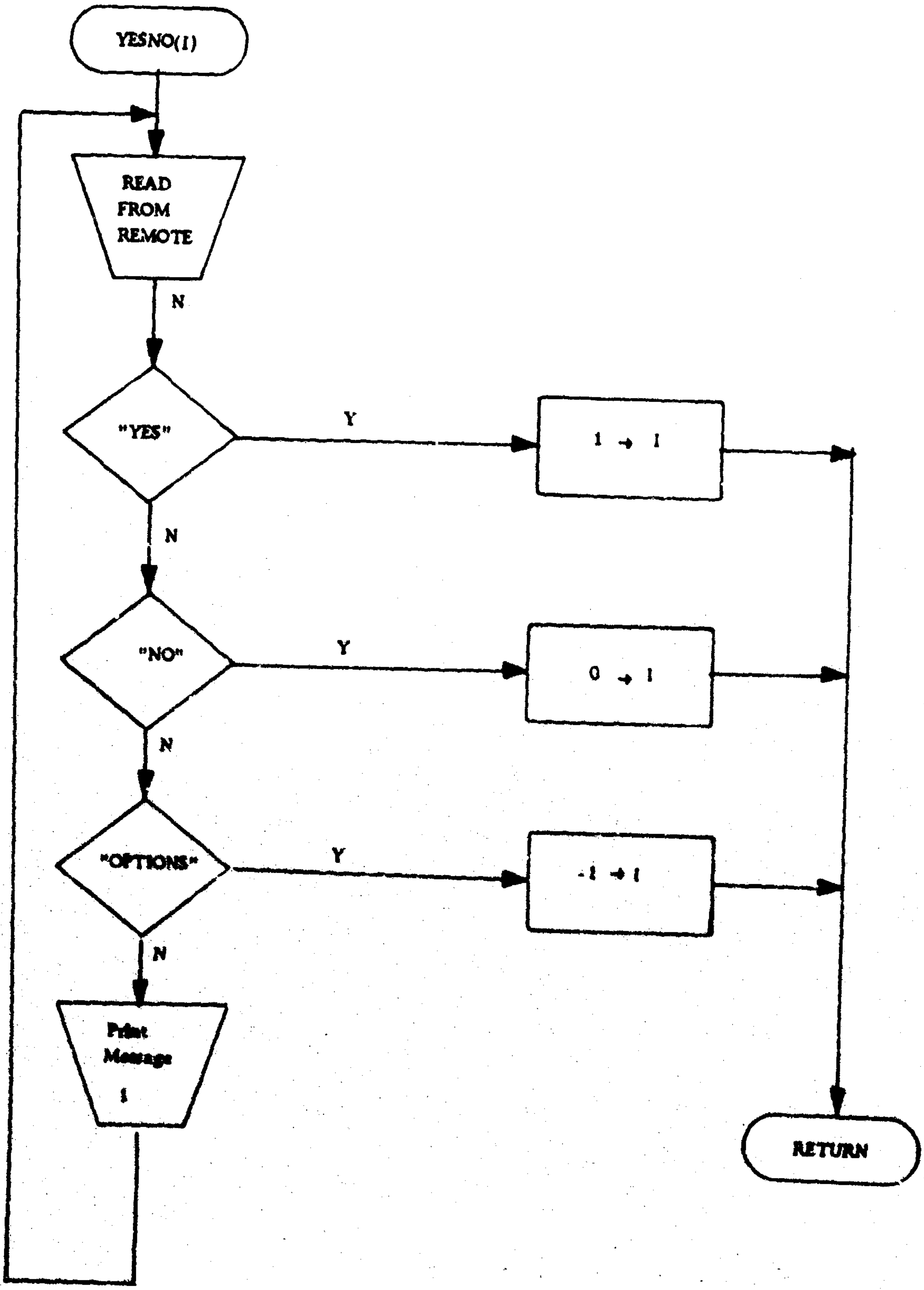


Figure IV-6. Subroutine YESNO

F1 = -1. The accession number of a single specified document is contained in ARG. The user may wish to specify more documents, so the calling program should return to NUMBER after processing the document specified in ARG.

F1 > 0. The user has specified a sequence of documents. The accession number of one of the documents is contained in ARG, and F1 = 1, 2, ... as ARG specifies the first, second, ... document in the sequence. The calling program should continue to return to NUMBER if additional documents in the sequence are required. NUMBER will return with F1 = 0 if the sequence is exhuasted. If the calling program is to terminate the sequence early, it can do so providing that the next call (for a new document or document sequence) specifies F1 = 0. This is generally good practice, since it allows different sections of the dialogue program to use differing and local names for the first argument of NUMBER.

The reason for giving the location of a retrieved document in a sequence (F1 > 0) is to avoid setting up counters in the calling program. For instance, when the bibliographic data of documents are being printed, it is desirable to stop and ask the user if any more documents are required after every fifth document. The statement IF (F1 - 5*(F1/5). EQ. 0) GO TO 100 tests this.

Figure IV-7 is the flowchart for NUMBER.

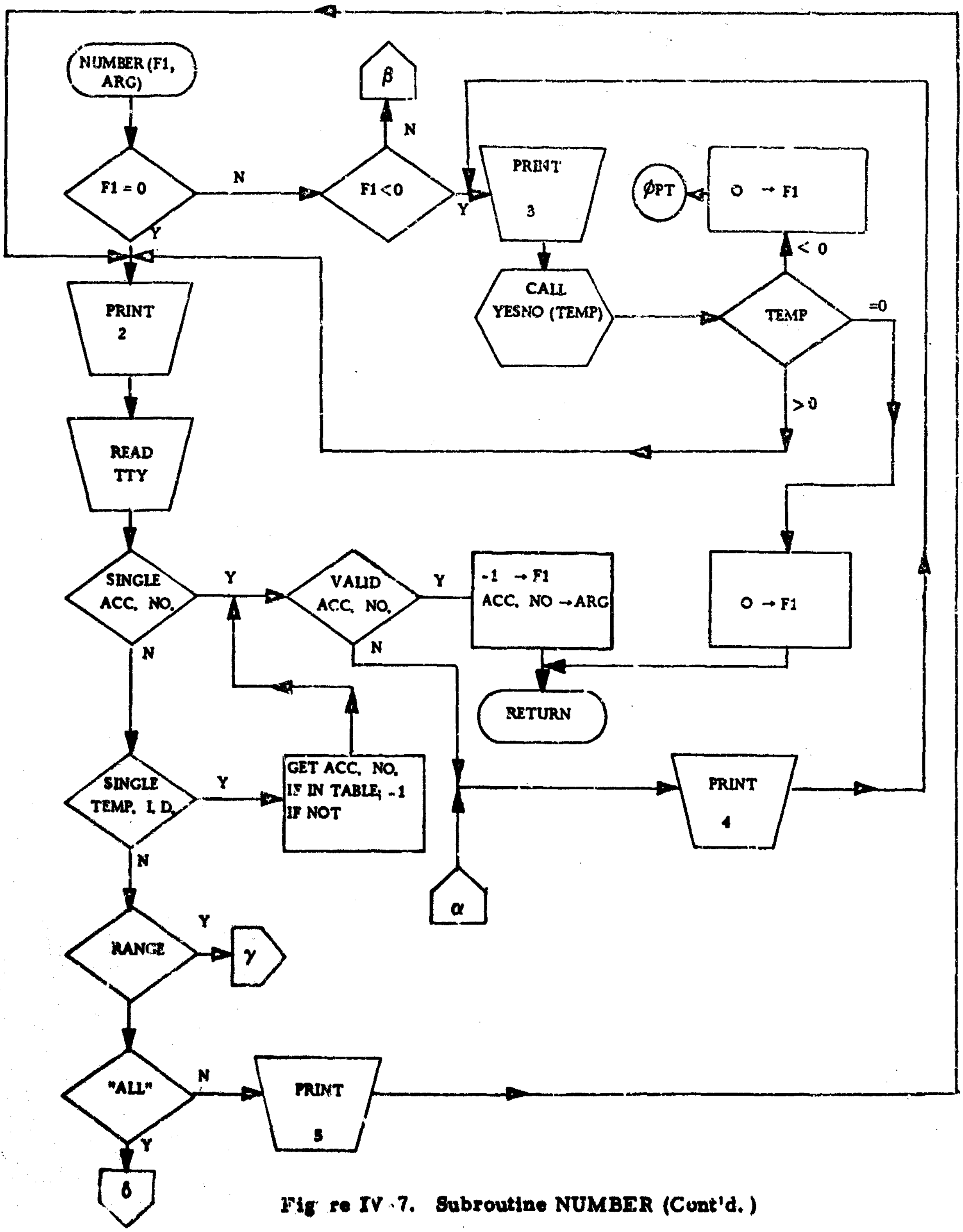


Fig re IV -7. Subroutine NUMBER (Cont'd.)

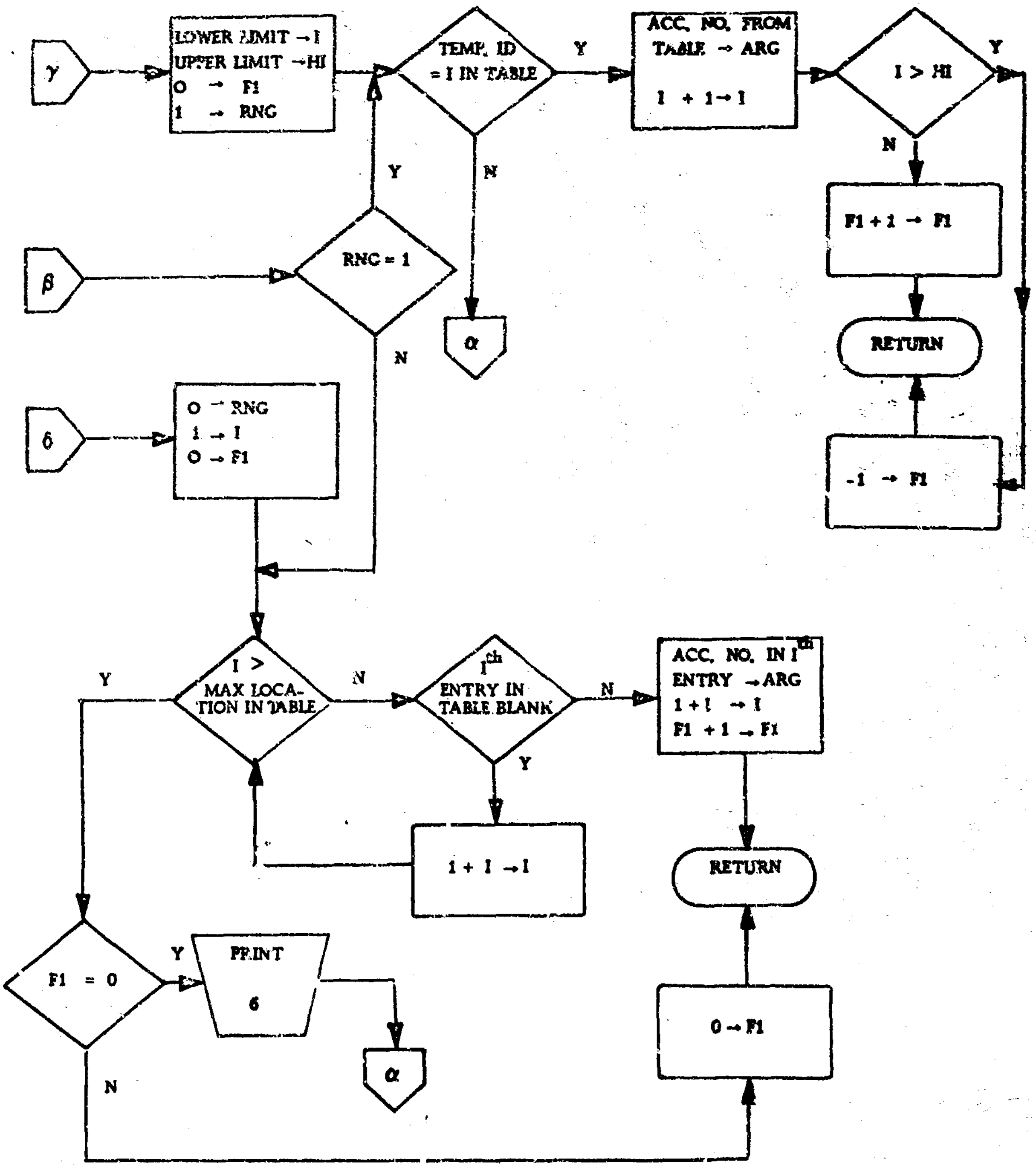


Figure IV-7. Subroutine NUMBER (Concluded)

SECTION V

STRUCTURE OF THE ON-LINE SYSTEM

This Chapter describes the overall structure of the on-line system. Implementation details are not included; they are covered in Chapter VI. Also excluded are the data preparation programs, which are discussed in Chapter VII.

V.1 Overall Structure

Figure V-1 shows the overall structure of the on-line system. Files are represented by symbols with rounded sides; rectangles represent programs. An arrow from A to B indicates that A calls B, if A and B are programs. If B is a file and A is a program, the arrow indicates that A writes on B; if A is a file and B is a program, then B reads from A.

The dialogue program keeps track of the status of the present query sequence by maintaining the query sequence status file. Because this file contains the information needed to direct the operation of the other programs in the on-line system, the dialogue program performs the executive function, and is resident in core at all times while the on-line system is in operation. For this reason, the core requirements of the dialogue program must be minimized; therefore, the only file that DIALOGUE will keep in core is the query sequence status file, which will contain the current query words, stems, concept numbers, weights, and various flags that specify the status of the query.

The four program modules that are loaded into core by the dialogue program are shown in Figure V-1 as the four blocks immediately below the dialogue program. Each of the program modules will be loaded with the subprograms that it calls. With one exception, CHOOSE, only one of the four program modules will be resident in core at once.

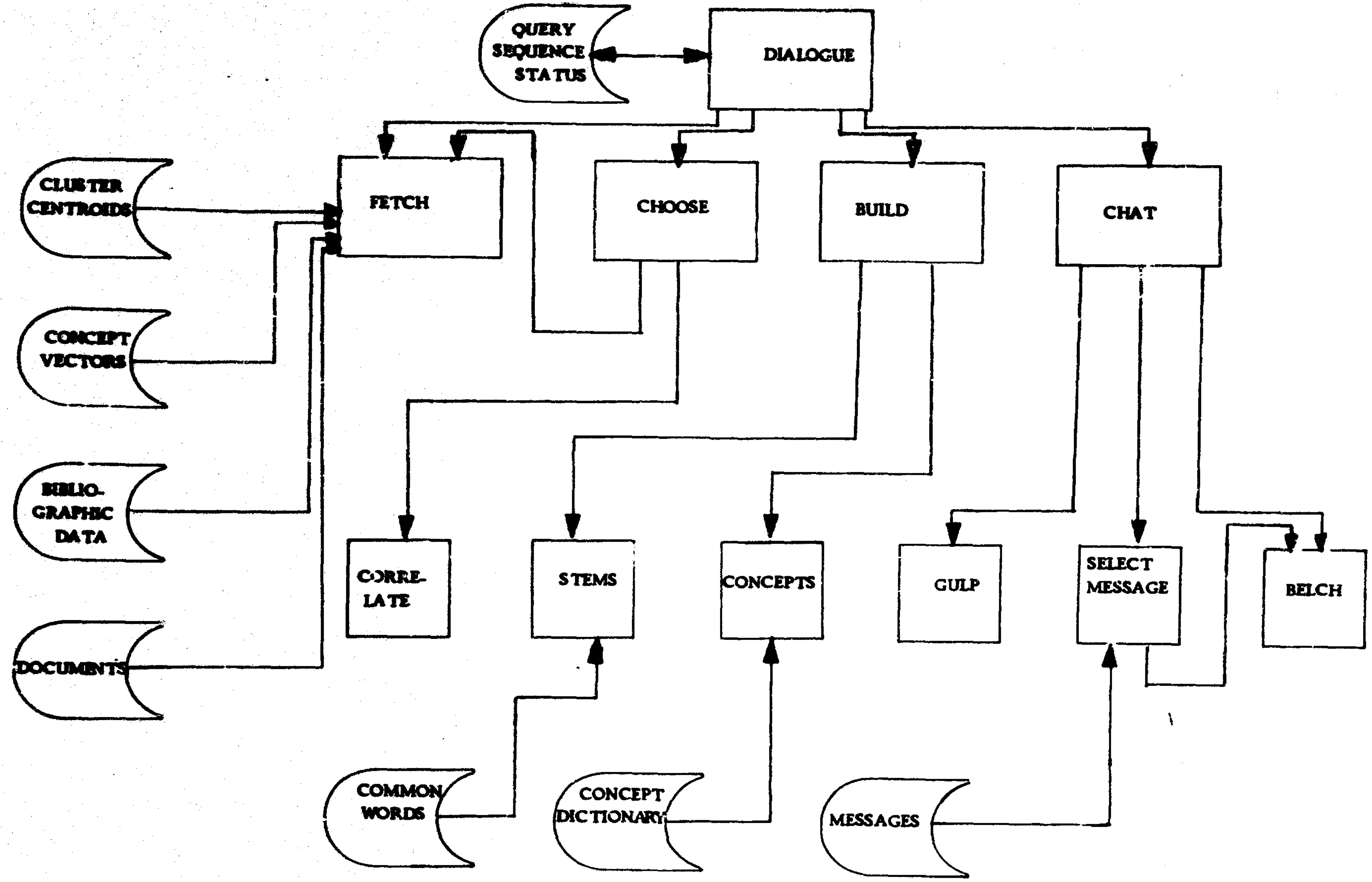


Figure V-1. On-Line System Structure

V.2 Files

The entities shown as files are not necessarily distinct files that will be stored on auxiliary storage devices; rather, every sizable data structure is identified here as a file so that an explicit decision concerning its residence can be made.

The four files shown on the left margin of Figure V-1 are arranged hierarchically in order of increasing minimum access time requirements. Exactly which file is resident of what type of auxiliary storage device is a decision to be based upon both the amount of auxiliary storage available and response time requirements. For the system to interact conversationally with the user, at least the cluster centroids and concept vectors must be stored on a high-speed direct-access storage device. The document file can be allocated to tape or disk storage. The programs that access the data base will be designed in a modular fashion to minimize the problems associated with reallocating the files among various storage media. However, there are fundamental programming differences between direct-access and sequential-access file manipulations, so a certain amount of reprogramming will be required to reallocate parts of the data base.

The remaining four files, namely QUERY SEQUENCE STATUS, COMMON WORDS, CONCEPT DICTIONARY, and MESSAGES, are each accessed by only one program module, and are small enough that they can easily be accommodated in core along with their using program modules. Therefore, these four files will be stored within their program modules, and will not appear as distinct GECOS III files in the On-Line Retrieval System.

The file structure has been designed to accommodate the widest possible variation in data base characteristics. The main contributor to this flexibility is the use of variable-length records in every file. This not only removes the need for some arbitrary limit on the size of each type record; it also greatly increases the efficiency with which the available disk storage space is used, because every record will occupy only the amount of space it requires*. The additional programming complexity introduced by the use of variable-length records will be well compensated.

The on-line files that will be accessed by program module FETCH are:

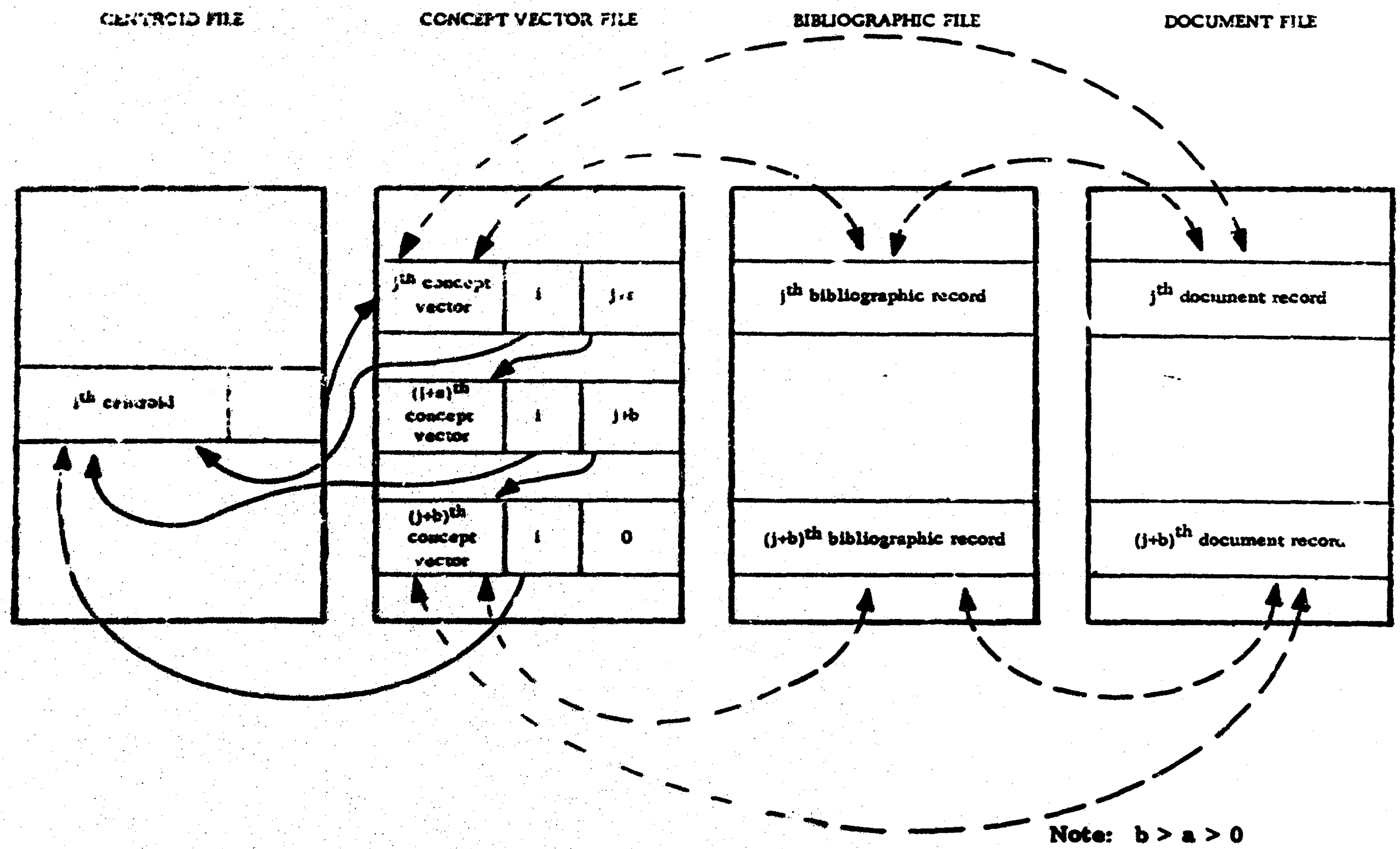
1) documents
2) bibliographic data
3) concept vectors
4) centroids

Before the on-line system can be used, these data must be loaded into four distinct GECOS III permanent files by SHOVEL, described in Chapter VI. Four separate files are used in order to permit all the records that are associated with a given document in the data base to be obtained by using only the accession number of the generating document. Because of this, no separate directory will be necessary, and cross-referencing from a concept vector to a bibliographic record to the document itself can be performed without intermediate accesses to a directory.

Figure V-2 illustrates the organization of the on-line files. The solid arrows represent an explicit "pointing" relationship; the dashed arrows represent an implicit "pointing" relationship that arises because the concept vector, bibliographic and document files are all ordered by accession number. Thus, the arrows indicate all the possible methods of cross-referencing the various files.

So that the files can be organized efficiently by accession number, it is required that the accession numbers be a compact set of positive integers starting at 1. If the data base is supplied

* Of course, if the data base as received consists of fixed-length records, use of the variable-length feature will be superfluous; however, it will nevertheless be incorporated into the system regardless of the data base characteristics, in order to make the file structure as flexible as possible.



Note: b > a > 0

Figure V-2. On-Line File Structures

without these integral accession numbers, it is a simple matter to number all the documents.

The use of a distinct file for each class of data also permits the selective loading of the various files. The On-Line System might be used for experi ments that would not access all of the files. In this case, the selective loading of the on-line files, by reducing disk usage, will increase operational economy beyond that which might otherwise be associated with experimentation with the full On-Line System. Any file can be loaded with its normal contents, or it can contain only a sentinel record that indicates that the file has not been loaded. Thus, if the user requests access to a file that has not been loaded, DIALOGUE can inform him that a file necessary for the operation he has requested has not been loaded. This is much more desirable than letting GECOS III abort the on-line system because of an illegal I/O request.

V. 2. 1 Documents

The document file will be stored one document per variable length record, ordered by accession number. This file will contain only the documents, and not bibliographic data, author, or citations.

V. 2. 2 Bibliographic Data

The bibliographic data file will contain one variable length record for each document in the document file, ordered in the same way as the document file. Each record will contain author and title information for one document.

V. 2. 3 Document Concept Vectors

The concept vector file will contain one concept vector per variable-length record, ordered by document accession number. Included in each record will also be a pointer to the next concept vector in each cluster to which the vector belongs. Each pointer consists of a cluster number and a document accession number. The cluster number identifies the cluster, and the accession number points to the next document in that cluster. The "end-of-cluster" indicator will be a cluster pointer with a zero accession number.

In Figure V-2, the three concept vectors shown are all members of the ith cluster. The cluster number is part of the pointer becanse the cluster-concept vector occupancy matrix is expected to be sparsely occupied. Organising the centroid file so that each concept vector points directly to the centroids of all its clusters provides the additional useful ability to obtain directly the centroid of all the clusters to which a given document belongs, given only the document's accession number. Obtaining centroids in this way will speed up document-document correlation.

V.2.4 Centroids

When the documents are clustered, one centroid will be generated for each cluster. All the centroids so generated will be stored in the centroid file, ordered by cluster number. Each centroid rocord will contain one cluster centroid and the accession number of the first concept vector in the cluster.

Although the centroid file is not required for the operation of the on-line system using the initial clustering technique (see Section III.4), its inclusion in the system design permits the system to be operated using other clustering techniques.

V.3 Program Modules

This Section introduces each program module with a brief description of its function. The details of implementation are covered in Section VIII.

V.3.1 DIALOGUE

DIALOGUE is described in Chapter IV, and therefore is omitted from this discussion. All other modules operate under the control of DIALOGUE.

V.3.2 FETCH

Program module FETCH performs all accesses to the on-line data base. Given a record number and a file designation, FETCH returns the record and size of the record. FETCH obtains only one record at a time; to obtain all the records in a file, FETCH must be called repeatedly.

FETCH will be loaded by itself or together with CHOOSE. FETCH will be loaded by itself when a file access is being performed that does not require selection of concept vectors based on their correlation with some query vector, such as when scanning of the bibliographic data or document file is taking place.

V.3.3 CHOOSE

Program module CHOOSE, given a query vector by DIALOGUE returns to DIALOGUE the accession numbers of the documents whose concept vectors have the highest correlation coefficients with the query vector. To do this, CHOOSE calls FETCH to obtain the centroids of all clusters, and then calls CORRELATE to determine which clusters to scan. When this is complete, FETCH is called to obtain the selected clusters, and the concept vectors in these clusters are similarly processed by CORRELATE. CHOOSE then returns to DIALOGUE the accession numbers of the documents whose concept vectors correlate most highly with the query.

V.3.4 BUILD

Program module BUILD operates on a list of words and produces a concept vector. It does this by first performing stem analysis by calling STEMS, then mapping the stems into concepts by calling CONCEPTS. Program STEMS includes within it the list of common words and the list of stems to be removed: program CONCEPTS includes within it the dictionary of content stems and the concept numbers and weights into which each is mapped.

Each word in a query can fall into one of three categories. It may be a common word that is deleted by STEMS, a word that generates a noncontent stem and, therefore, is not mapped into a concept, or a word that generates a content stem and therefore is mapped into one or more concepts. BUILD will recognize and differentiate between these three cases, and report this information to DIALOGUE along with the generated concept vector and stems.

BUILD will be called to process a query before calling CHOOSE. When document-document correlation is being performed, BUILD will not be used, since the query vector in that case will be obtained by using FETCH to access the concept vector file.

V.3.5 CHAT

Program module CHAT communicates with the user. Standard On-Line System messages are sent to the user by calling SELECT. Given a message number, SELECT accesses the file of messages, selects one, and calls BELCH to transmit the message. BELCH transmits one line to the remote terminal; GULP reads a line from the terminal.

When documents are being printed at the remote terminal, SELECT will not be used. DIALOGUE will obtain the data to be sent by calling FETCH, and then call BELCH to transmit. Data obtained from other program modules, such as BUILD, will also be transmitted without a call to SELECT.

V.4 Example

Figure V-3 illustrates the roles played by the various program modules by showing the sequence of events that might take place during the processing of a query. This example shows only the gross features of query processing and document-document correlation; a sophisticated user would cause a much more complex process to take place.

During operation of the system, DIALOGUE performs a function in addition to those shown explicitly in the flowchart; it directs the loading of the other prc am modules.

The user begins the sequence by entering a query, which is read by CHAT. BUILD is then loaded, and performs stem and concept analysis, producing a concept vector if the query contains any words that generate content stems. DIALOGUE stores this concept vector as the query vector, and loads CHOOSE and FETCH together. By calling FETCH and CORRELATE, CHOOSE determines the accession numbers of the documents whose concept vectors correlate most highly with the query vector. This list is passed to DIALOGUE.

When DIALOGUE has received the query results, it loads CHAT to transmit the results to the user. At this point, the user might elect to enter a new query, in which case DIALOGUE clears QUERY SEQUENCE STATUS, or he might elect document-document correlation. He also has several other options, which are not shown in this example.

Document-document correlation is performed by using FETCH to obtain the concept vectors that are to be used as query vectors, and then calling CHOOSE in the same fashion as when processing a user-generated plain text query.

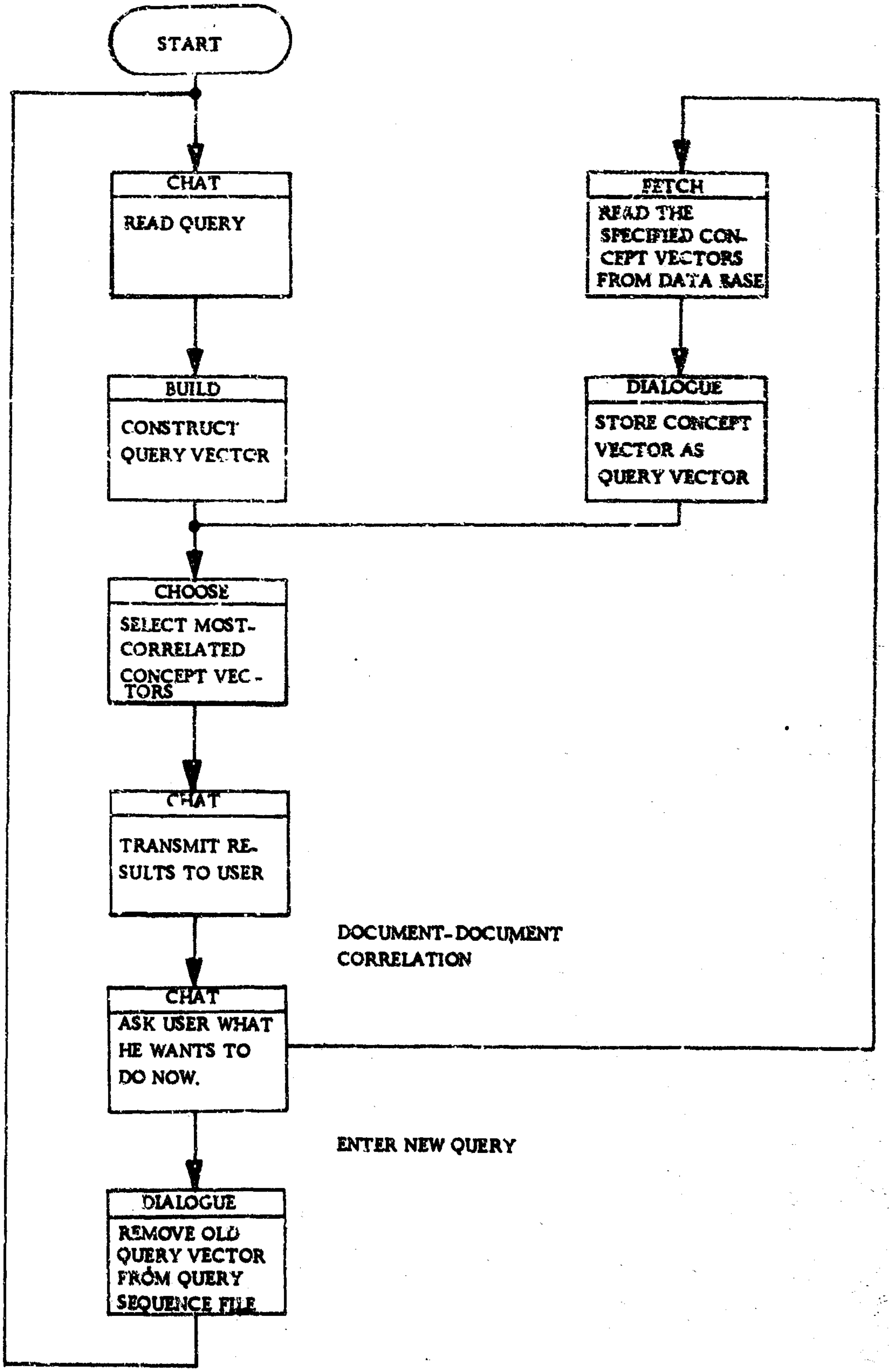

START
CHAT
READ QUERY
BUILD
CONSTRUCT QUERY VECTOR
CHOOSE
SELECT MOST-CORRELATED CONCEPT VEC-TORS
CHAT
TRANSMIT RE-SULTS TO USER
CHAT
ASK USER WHAT HE WANTS TO DO NOW.
DOCUMENT-DOCUMENT CORRELATION
ENTER NEW QUERY
DIALOGUE
REMOVE OLD QUERY VECTOR FROM QUERY SEQUENCE FILE
FETCH
READ THE SPECIFIED CON-CEPT VECTORS FROM DATA BASE
DIALOGUE
STORE CONCEPT VECTOR AS QUERY VECTOR

Figure V-3. Example of System Operation

SECTION VI

DATA PREPARATION PROGRAMS - IMPLEMENTATION DETAILS

This Chapter describes the implementation of the programs which load the on-line files and prepare the data for loading the files. The rationale for each of these programs is given in Section III; this Chapter is concerned only with the details of implementation. The discussion is divided into three sections: concordance and microconcordance preparation, dictionary and concept vector generation, and loading the on-line files.

VI. 1 Concordance Preparation

Figure III-1 illustrates the concordance preparation process. This discussion treats the three operations that are shown as rectangles in Figure III-1: microconcordance preparation, sort and merge.

VI. 1. 1 Microconcordance Preparation and Stem Analysis

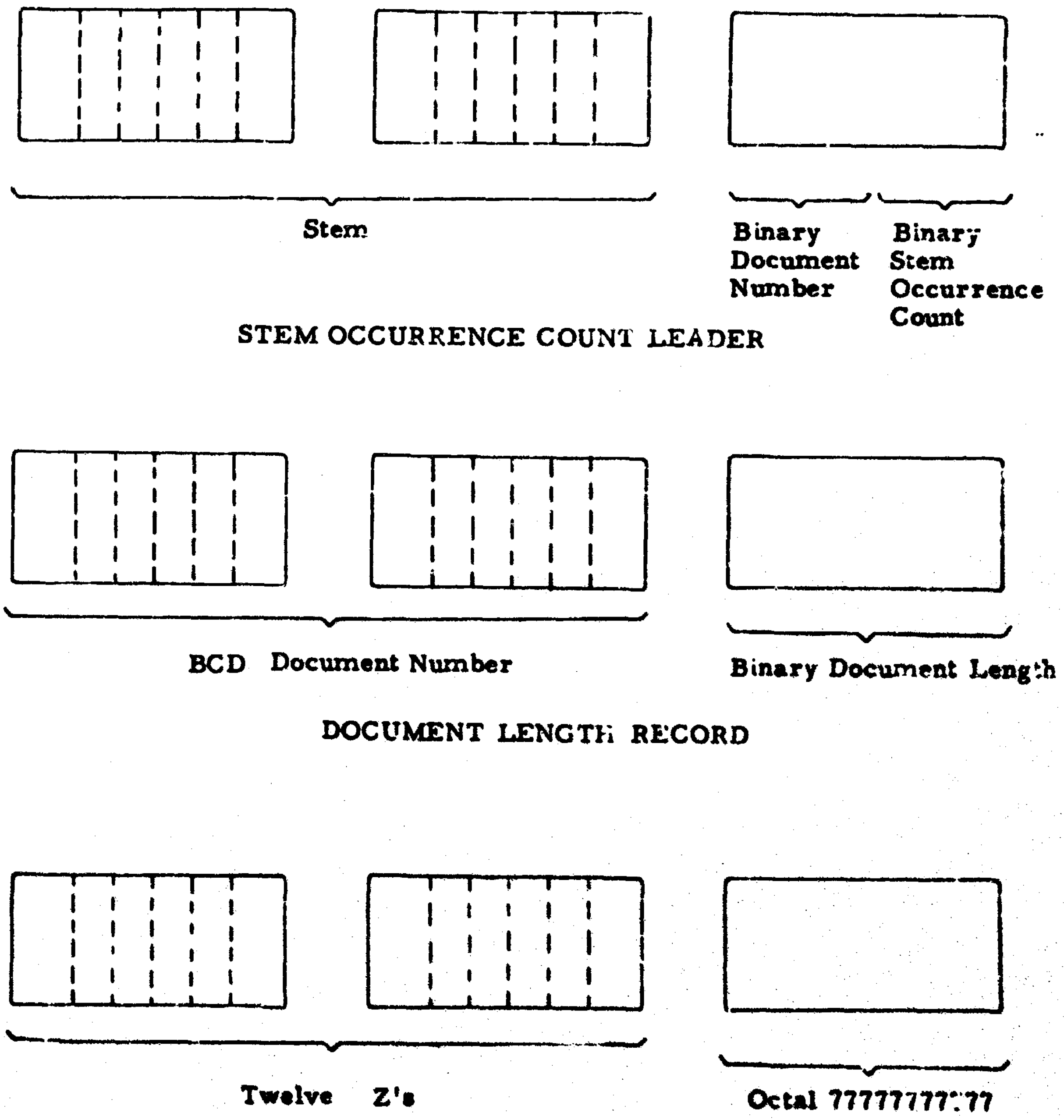
This discussion first considers the contents of the microconcordance tape, and then its production. Figure VI-1 is a Backus Normal Form (BNF) specification of the content of the microconcordance tape; Figures VI-2, VI-3, and VI-4 illustrate the BNF specification with diagrams.

The various physical record types that make up the microconcordance tape are diagrammed in Figure VI-2. A stem occurrence count leader is a stem, represented in two words as twelve characters, followed by a document-count pair, which gives the number of times the stem occurs in the specified document. A document length record indicates the number of stems counted in the document whose number is specified by twelve characters. The sentinel record is used in this case to indicate the

```
<microconcordance tape>              ::=   <microconcordance>† <sentinel>
<sentinel>                           ::=   <Z>^12 <Octal 777777777777>
<microconcordance>                   ::=   <stem occurrence count leader>† <document length record>
<stem occurrence count leader>       ::=   <stem> <document-count pair>
<document length record>             ::=   <BCD document number> <binary document length>
<stem>                               ::= | <letter>^12
                                         | <letter>^11 <blank>   | <letter>^10 <blank>^2 | <letter>^9 <blank>^3
                                         | <letter>^8 <blank>^4  | <letter>^7 <blank>^5  | <letter>^6 <blank>^6
                                         | <letter>^5 <blank>^7  | <letter>^4 <blank>^8  | <letter>^3 <blank>^9
                                         | <letter>^2 <blank>^10 | <letter>^1 <blank>^11 |
<document-count pair>                ::=   <binary document number> <binary stem occurrence count>
<BCD document number>                ::=   <BCD number>
<binary document length>             ::=   <binary number>
<letter>                             ::=    A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<blank>                              ::=
<binary document number>             ::=   <binary number>
<binary stem occurrence count>       ::=   <binary number>
<BCD number>                         ::=   <decimal digit>†
<binary number>                      ::=   <binary digit>†
<decimal digit>                      ::=    0|1|2|3|4|5|6|7|8|9
<binary digit>                       ::=    0|1

NOTE:                 <s>†           ::=   <s>| <s>† <s>
                      <s>^n          ::=   <s>^n-1 <s>
                      <s>^1          ::=   <s>
```

Figure VI-1. BNF Specification of Microconcordance Tape



STEM OCCURRENCE COUNT LEADER

DOCUMENT LENGTH RECORD

SENTINEL

Figure VI-2. Microconcordance Physical Record Types

end of tape. The sentinel record is composed of twelve Z characters followed by a word containing a large number.

A microconcordance is a concordance of one document; a microconcordance can be constructed in core, in contrast to the entire concordance, whose production must use a tape sort. Each microconcordance is constructed after reading one document; it includes an occurrence count for each stem in the document and the total number of stems in the document. A single microconcordance is diagrammed in Figure VI-3. The tape of microconcordances, diagrammed in Figure VI-4, contains a microconcordance of each document, followed by one sentinel record.

Figure VI-5 is a macro flowchart of the microconcordance generation process. A document is read and analyzed one word at a time. Subprogram GRAB obtains one word of text; after incrementing the document length count, each obtained word is compared to a list of common words, containing articles, conjunctions, and prepositions (see Figure VI-10); if the stem is one of these words, it is dropped and another word from the document is GRABbed. If the word is not a common word, routine STEM is called to obtain the stem of the word. The stem is compared to the list of stems. If the stem has already been found in this document, its occurrence count is incremented; if the stem has not yet been encountered in this document, it is added to the list. Another word in the document is then GRABbed. This process continues, until all the words in the document have been GRABbed; at this time, the list of stems and their frequencies are written on tape as stem occurrence count leaders, and the document length count is written as a document length record.

A microconcordance is generated for each document in the data base. When the data base is exhausted, a sentinel record is written on the microconcordance tape, statistics summarizing the run are printed, and processing is terminated.

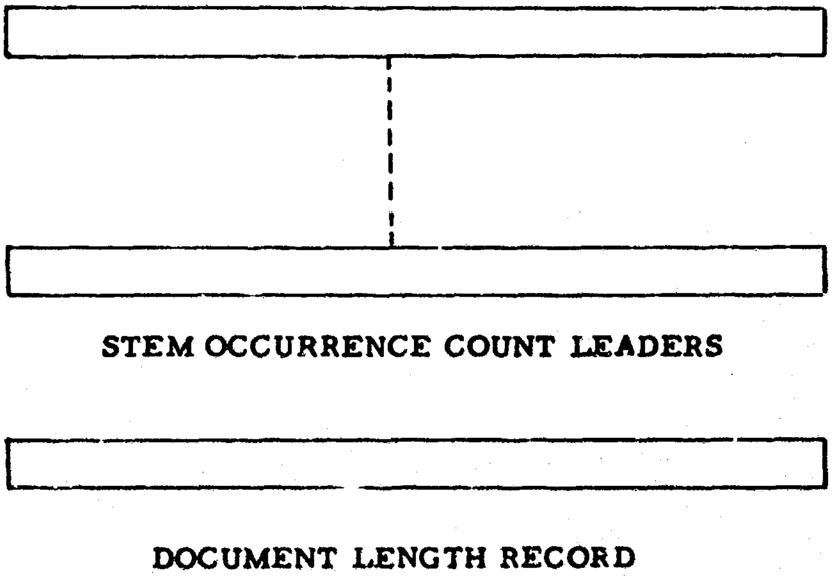


Figure VI-3. Microconcordances

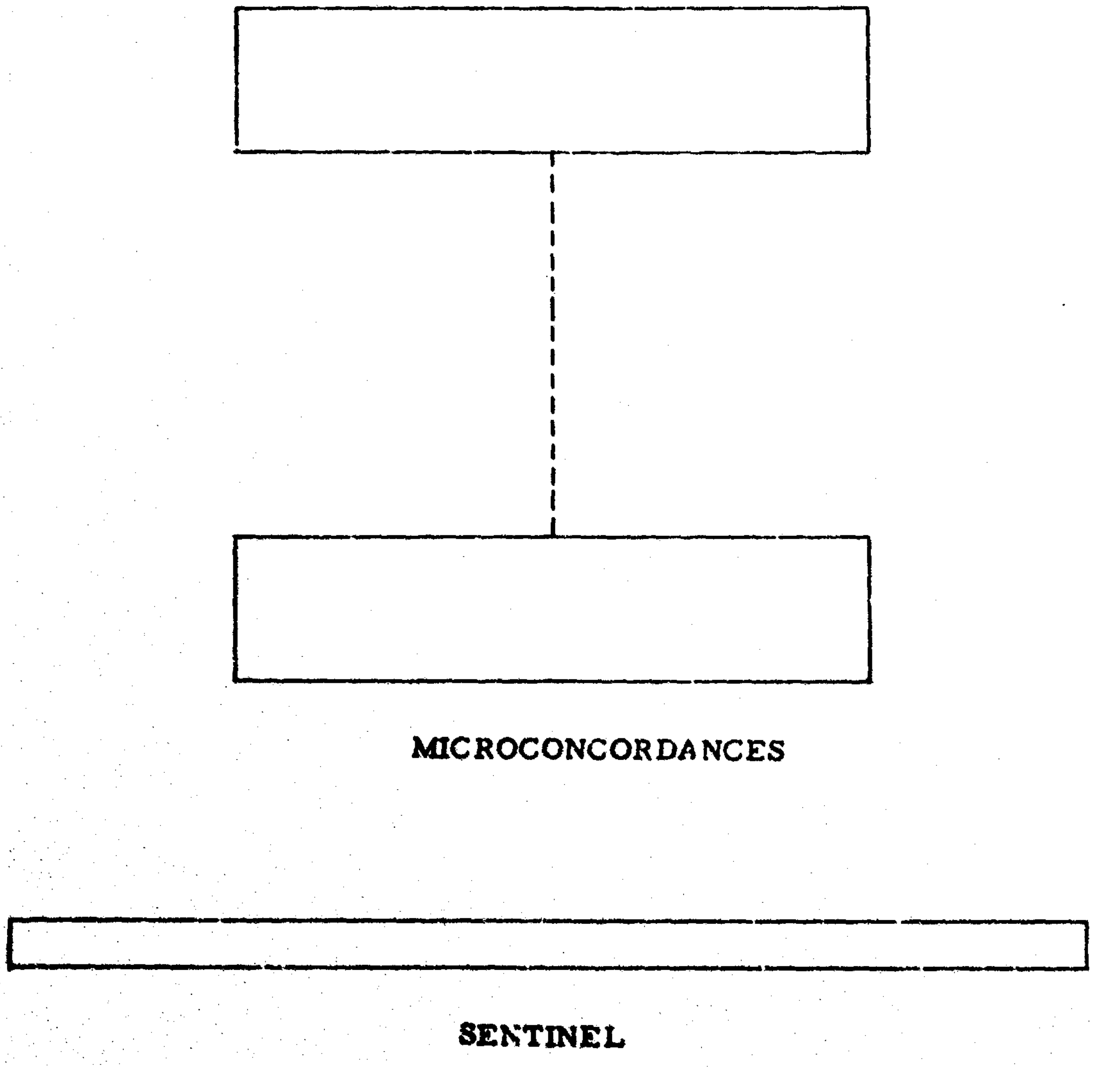


Figure VI-4. Microconcordance Tape Format

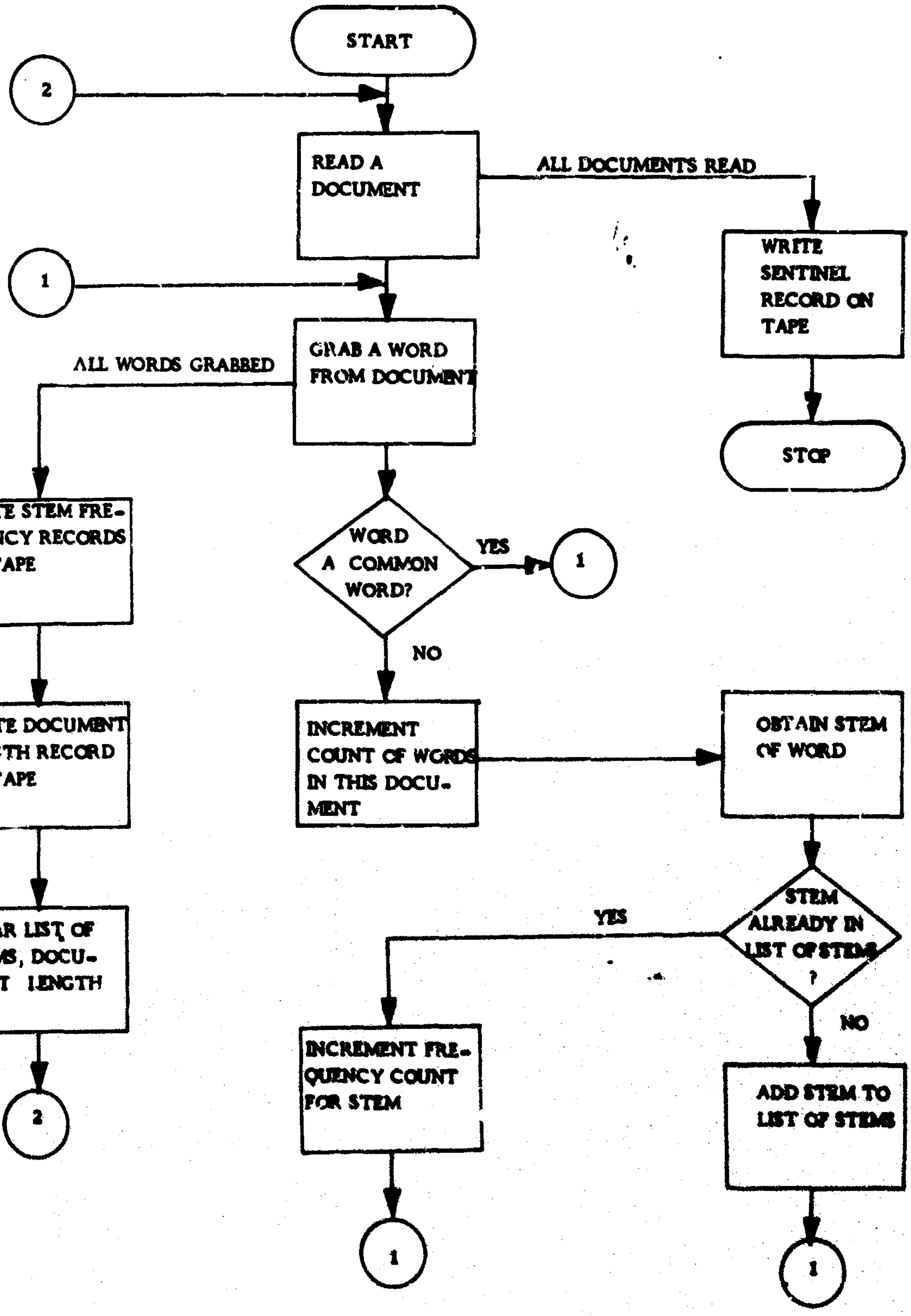

START
2
READ A DOCUMENT
ALL DOCUMENTS READ
WRITE SENTINEL RECORD ON TAPE
STOP
1
GRAB A WORD FROM DOCUMENT
ALL WORDS GRABBED
WRITE STEM FREQUENCY RECORDS ON TAPE
WORD A COMMON WORD?
YES
1
NO
WRITE DOCUMENT LENGTH RECORD ON TAPE
INCREMENT COUNT OF WORDS IN THIS DOCUMENT
OBTAIN STEM OF WORD
STEM ALREADY IN LIST OF STEMS ?
YES
NO
CLEAR LIST OF STEMS, DOCUMENT LENGTH
INCREMENT FREQUENCY COUNT FOR STEM
ADD STEM TO LIST OF STEMS
2
1
1

Figure VI-5. Microconcordance Generation: Macro Flowchart

Figure VI-6 is a flowchart of the operation of subprogram GRAB. GRAB will have two input parameters, DOCLENGTH and DOC. DOCLENGTH is an integer variable that specifies the number of characters in array DOC, which contains the document being processed. GRAB will return the output parameters LENGTH, an integer variable specifying the number of characters in the detected word, and WORD, an array of characters containing the word.

The operation of GRAB is very straightforward. It simply looks for and returns as word strings successive alphabetic characters. So that contractions will not be treated as two words, the apostrophe will be included in the list of alphabetic characters.

Figure VI-7 is a flowchart of the operation of subprogram STEM. STEM will have one pair of parameters, WORD and LENGTH, that will serve as input and output parameters. When STEM is called, WORD, an array of characters, will contain the word to be stem-analyzed in its first LENGTH positions. STEM will return the stem of the word in array WORD, decreasing LENGTH as necessary.

The list of suffixes that will be removed is shown in Figure VI-8. Because of the two-pass algorithm that will be used for stem analysis, no suffixes longer than four letters need to be considered. Compound suffixes will be removed as two separate steps; for example, "permanently" will be shortened on the first pass to "permanent", and on the second pass to "perman". Thus, "permanently" will be mapped into the same stem as "permanence".

VI.1.2 Sort

The microconcordance tape contains occurrence-weight pairs for all words occurring in the data base, ordered by document number of each occurrence. The document lengths are interspersed with occurrence

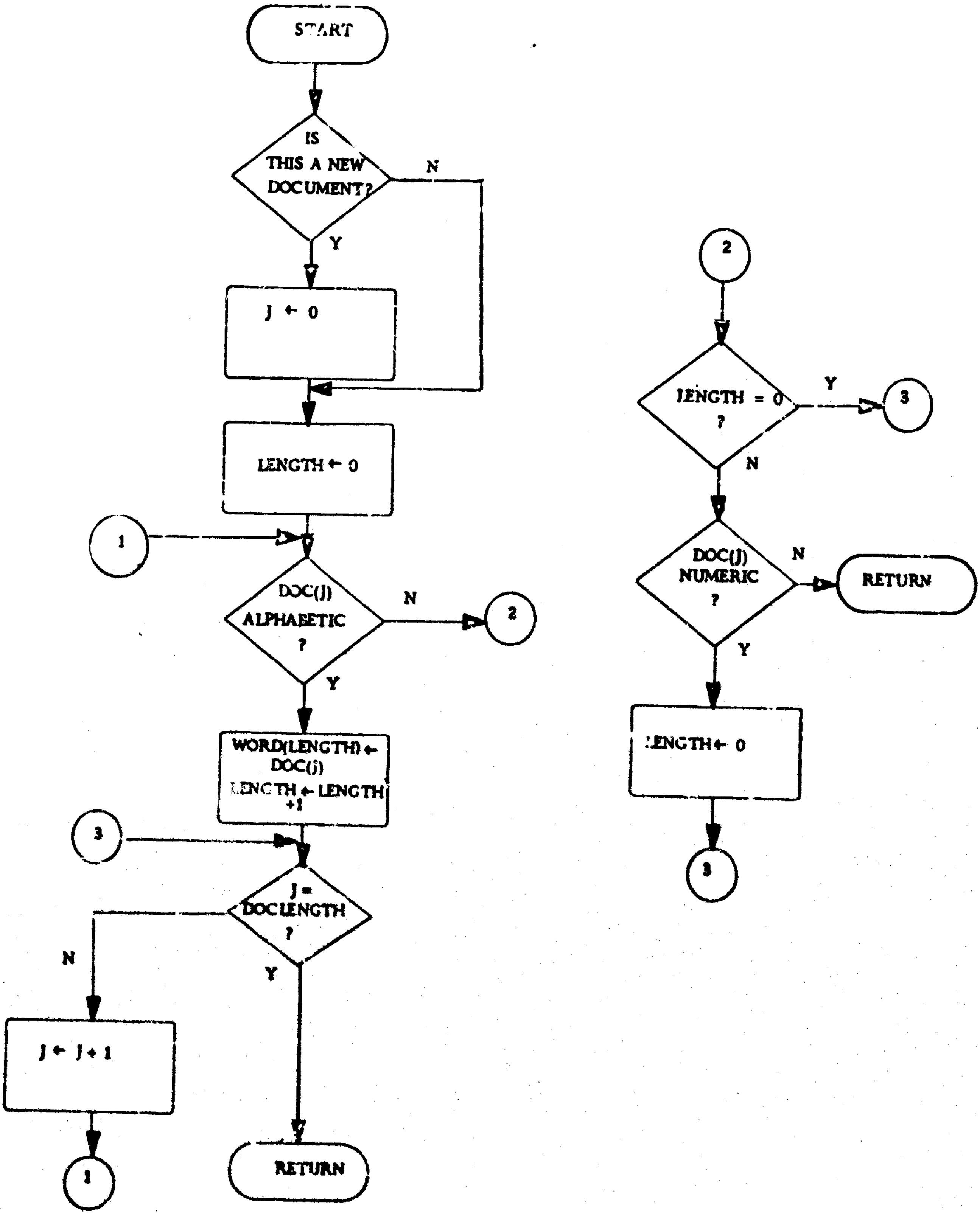


Figure VI-6. Subprogram GRAB (DOCLENGTH, DOC, LENGTH WORD)

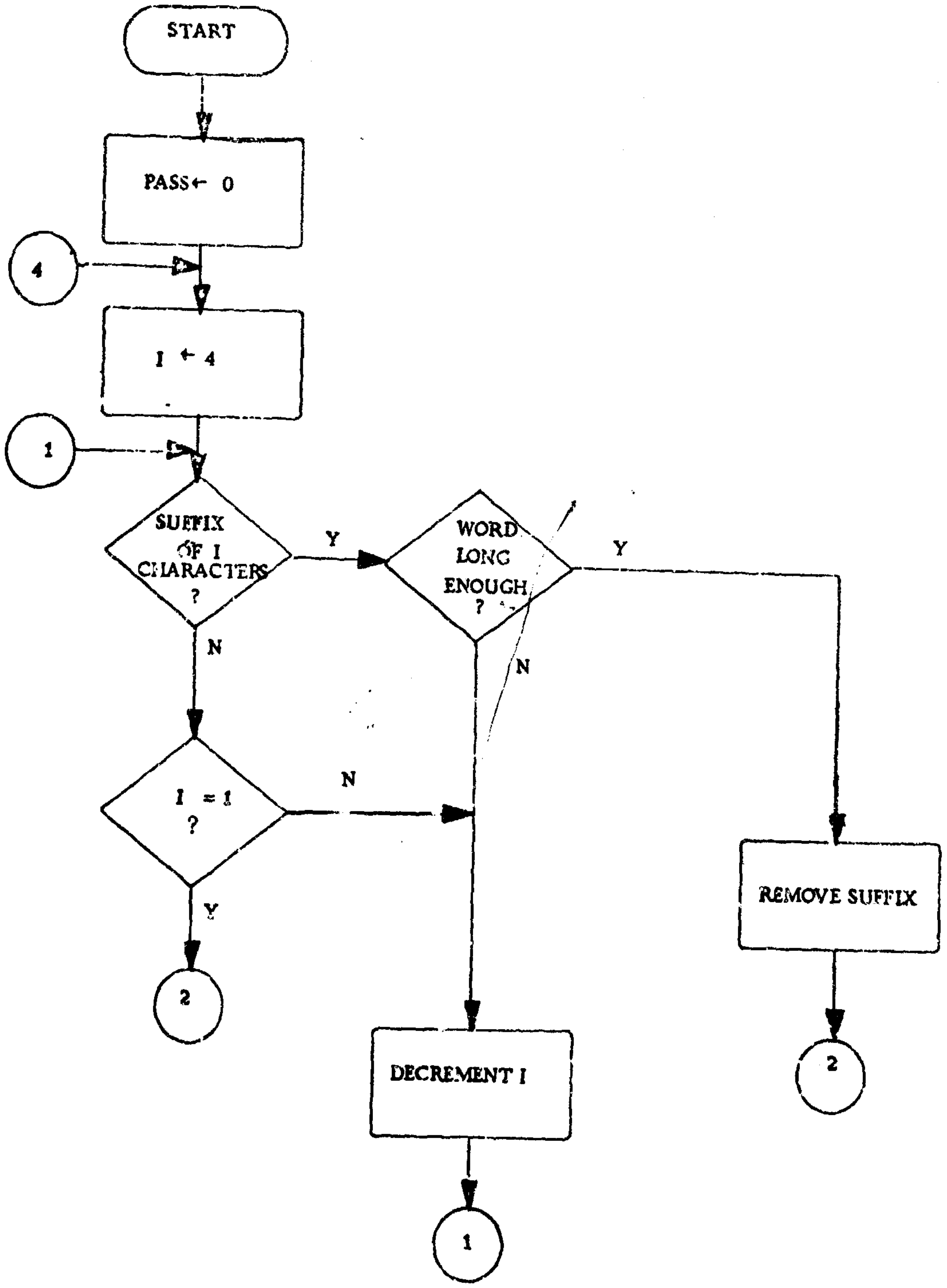


Figure VI-7. Subprogram STEM (WORD, LENGTH) (Cont'd.)

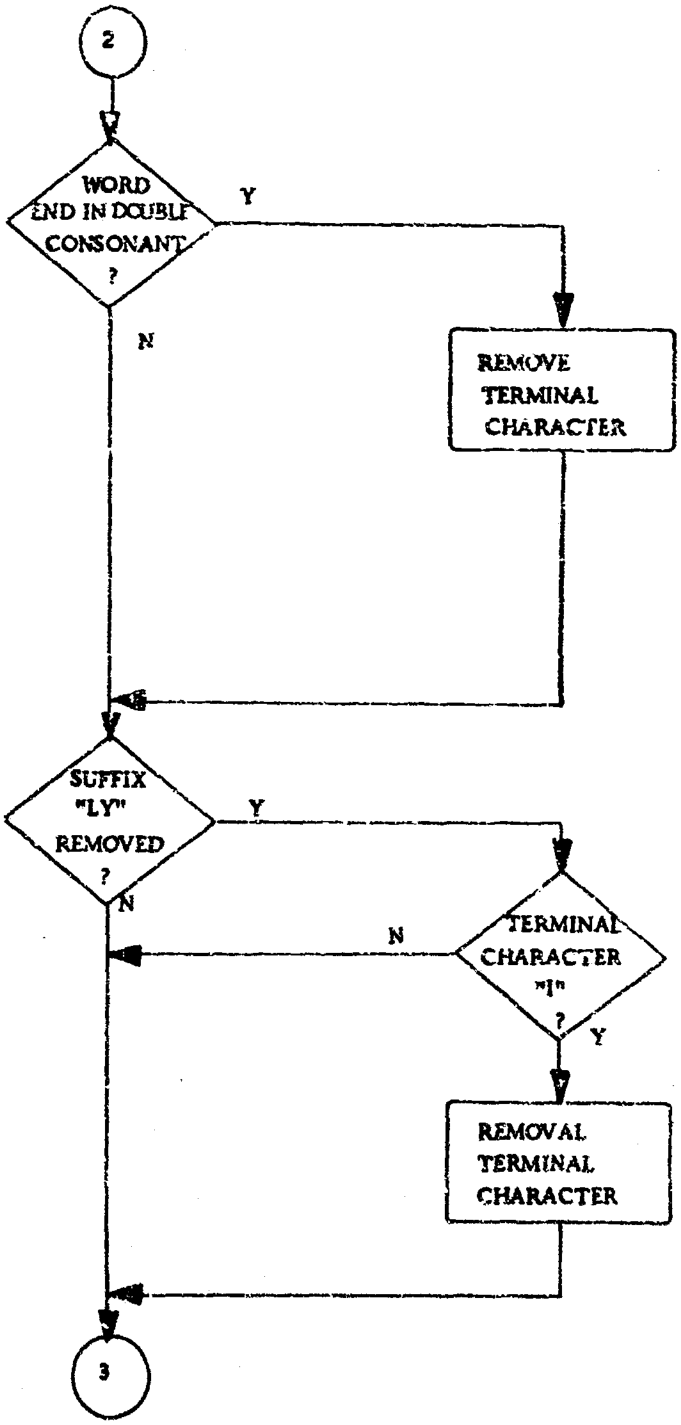

2
WORD END IN DOUBLE CONSONANT ?
Y
N
REMOVE TERMINAL CHARACTER
SUFFIX "LY" REMOVED ?
Y
N
TERMINAL CHARACTER "I" ?
N
Y
REMOVAL TERMINAL CHARACTER
3

Figure VI-7. Subprogram STEM (WORD, LENGTH) (Cont'd.)

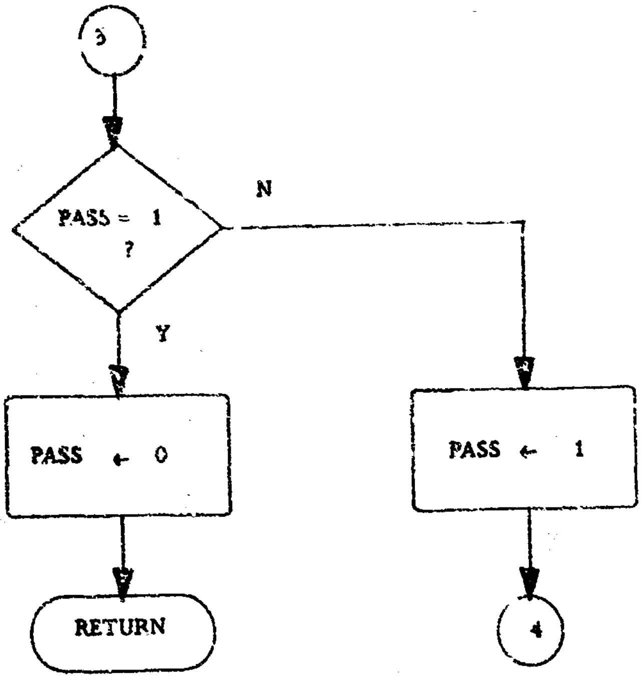


Figure VI-7. Subprogram STEM (WORD, LENGTH) (Concluded)

4 letters

able
ance
cant
cide
duce
ence
ever
hand
ient
itie
lent

lert
less
mate
ment
mity
ness
pear
sert
ship
sing

sorb
teen
ther
tial
tion
tize
turb
vent
vice
wise

3 letters

age
ant
ary
ate
bar
can
der
eed
ent
est
eth
ful
gen
ial

ian
ied
ies
ing
ish
ism
ion
ist
ity
ive
ize
lay
iel
man
men
not

ode
ose
ous
ply
sty
tal
ter
tic
tle
ule
ure
var
way

2 letters

's
al
an
ar
cy
ed
en
er

et
ic
ly
on
or
ou
ry
s'
th

1 letter

'
e
s
y

Figure VI-8. Suffixes

records. For the processing steps that follow, it is necessary to have a concordance ordered first by stem and second by document. It is also desired that the document lengths all be located together at the beginning of the tape, ordered by document number. Obviously, then, the tape of microconcordances must be sorted.

The sort will be performed by using the GE 625/635 Sort/Merge Program. The records will be sorted in ascending order; the key will be the first two words of each record. A glance at Figure VI-2 will reveal that the document length records will all sort together at the beginning of the tape; these will be followed by the stem occurrence count leaders, ordered by stem; the sentinel will sort to the end of the tape.

The 625/635 Sort/Merge Program includes an option that will reduce the time required to perform the sort. It is possible to choose what action the sort program will perform when it encounters input records with identical keys. This option will be used to keep the records of equal key in the order in which they occurred in the original file; its use will eliminate the need for specifying a secondary key. The document lengths will appear sorted by document, because they are ordered by document number in the microconcordance tape; similarly, the stem occurrence count leaders will appear sorted first by stem, and second by document number, because they too are ordered by document number in the microconcordance tape. The use of the ELECT feature in this way permits the size of the key field to be reduced by 1/3, which will greatly reduce the CPU time required to perform a sort.

VI.1.3 Merge

The merge program is the final step in the production of the concordance. The output tape produced by the sort contains a stem occurrence leader for each stem occurrence in each document. The merge program combines these, producing for each stem one stem occurrence count leader followed by a sentinel. Figure VI-10 is a BNF specification

about
above
across
after
against
all
almost
alone
along
also
although
always
among
am
and
another
an
anybody
anyone
any
anything
anywhere
apart
are
around
a
aside
as
at
away
awfully
because
been
before
behind
being
below
be
between
beyond
both
but
by
cannot
can
could
did

does
doing
done
do
down
during
each
either
else
elsewhere
enough
etc
even
ever
everyone
every
everything
everywhere
except
few
for
forth
from
furthermore
get
gets
got
had
hardly
has
have
having
hence
herein
here
her
herself
he
him
himself
his
hither
howbeit
however
how
if
inasmuch

indeed
inner
in
insofar
instead
into
inward
I
is
it
itself
its
just
keep
kept
least
less
lest
many
may
me
might
mine
moreover
more
most
much
must
my
myself
neither
nevertheless
next
nobody
none
nor
no
nothing
not
nowhere
of
oh
one
ones
only
on
onto

or
other
others
otherwise
ought
our
ourselves
ours
outside
over
own
per
please
plus
quite
rather
really
right
self
selves
several
shall
she
should
since
six
somebody
some
something
sometimes
somewhat
so
still
such
ten
than
that
their
theirs
them
themselves
thence
then
thereby
therefore
there
the

Figure VI-9. Common Words

these
they
this
those
though
throughout
thus
together
too
to
toward
two
underneath
under
unless
until
unto
upon
up
upward
us
very
was
well
were
we
whatever
what
whence
whenever
when
where
wherever
whether
which
while
whom
who
whose
why
will
within
without
with
would
yes

yet
your
yourself
yourselves
yours
you

Figure VI-9. Common Words (Concluded)

```
<concordance>                        :: =  <document length record>^† <single stem concordance>^† <sentinel>
<document length record>             :: =  <BCD document number>  <binary document length>
<single stem concordance>            :: =  <stem occurrence count leader>   <stem occurrence count trailer>*
                                           <sentinel>
<sentinel>                           :: =  <Z>^12 <Octal 777777777777>
<BCD document number>                :: =  <BCD number>
<binary document length>             :: =  <binary number>
<stem occurrence count leader>       :: =  <stem> <document-count pair>
<stem occurrence count trailer>      :: =  <document-count pair>^3
                                          |<document count pair>^2  <binary zero>
                                          |<document-count pair>  <binary zero>^2
<BCD number>                         :: =  <decimal digit>^†
<binary number>                      :: =  <binary digit>^†
<stem>                               :: =  <letter>^12
                                          |<letter>^11  <blank>   |<letter>^10  <blank>^2  |<letter>^9  <blank>^3
                                          |<letter>^8   <blank>^4 |<letter>^7   <blank>^5  |<letter>^6  <blank>^6
                                          |<letter>^5   <blank>^7 |<letter>^4   <blank>^8  |<letter>^3  <blank>^9
                                          |<letter>^2   <blank>^10 |<letter>    <blank>^11
<document-count pair>                :: =  <binary document number>  <binary stem occurrence count>
```

Figure VI-10. BNF Specification of Concordance (Cont'd.)

```
<binary document number>            :: =   <binary number>
<binary stem occurrence count>      :: =   <binary number>
<decimal digit>                     :: =    0|1|2|3|4|5|6|7|8|9
<binary digit>                      :: =    0|1
<letter>                            :: =    A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<blank>                             :: =
```

NOTE:

$<s>^{\dagger}$ :: = $<s> \mid <s>^{\dagger}$ <s>

$<s>^{n}$ :: = $<s>^{n-1}$ <s>

$<s>^{1}$ :: = <s>

$<s>^{*}$ :: = $\phi \mid <s>^{*}$ <s>

Figure VI-10. BNF Specification of Concordance

of the concordance tape; Figures VI-11, VI-12, and VI-13 illustrate the specification with diagrams.

Figure VI-11 shows the various physical record types that make up the concordance. Note that those are the same record types in the microconcordance tape, with the addition of the stem occurrence count trailer. This record contains three document number-stem occurrence count pairs, enabling it to convey the information carried by three stem occurrence count leaders. Figure VI-12 is a diagram of the concordance information produced by the merge program for one stem; Figure VI-13 shows how single-stem concordances are combined to form the concordance tape.

Figure VI-14 is a flowchart of the merge process. Inspection of the flowchart will reveal that the program simply combines stem occurrence count leaders with identical stems into the single-stem concordances shown in Figure VI-12. It is expected that the merge process will reduce the size of the concordance by at least 40%.

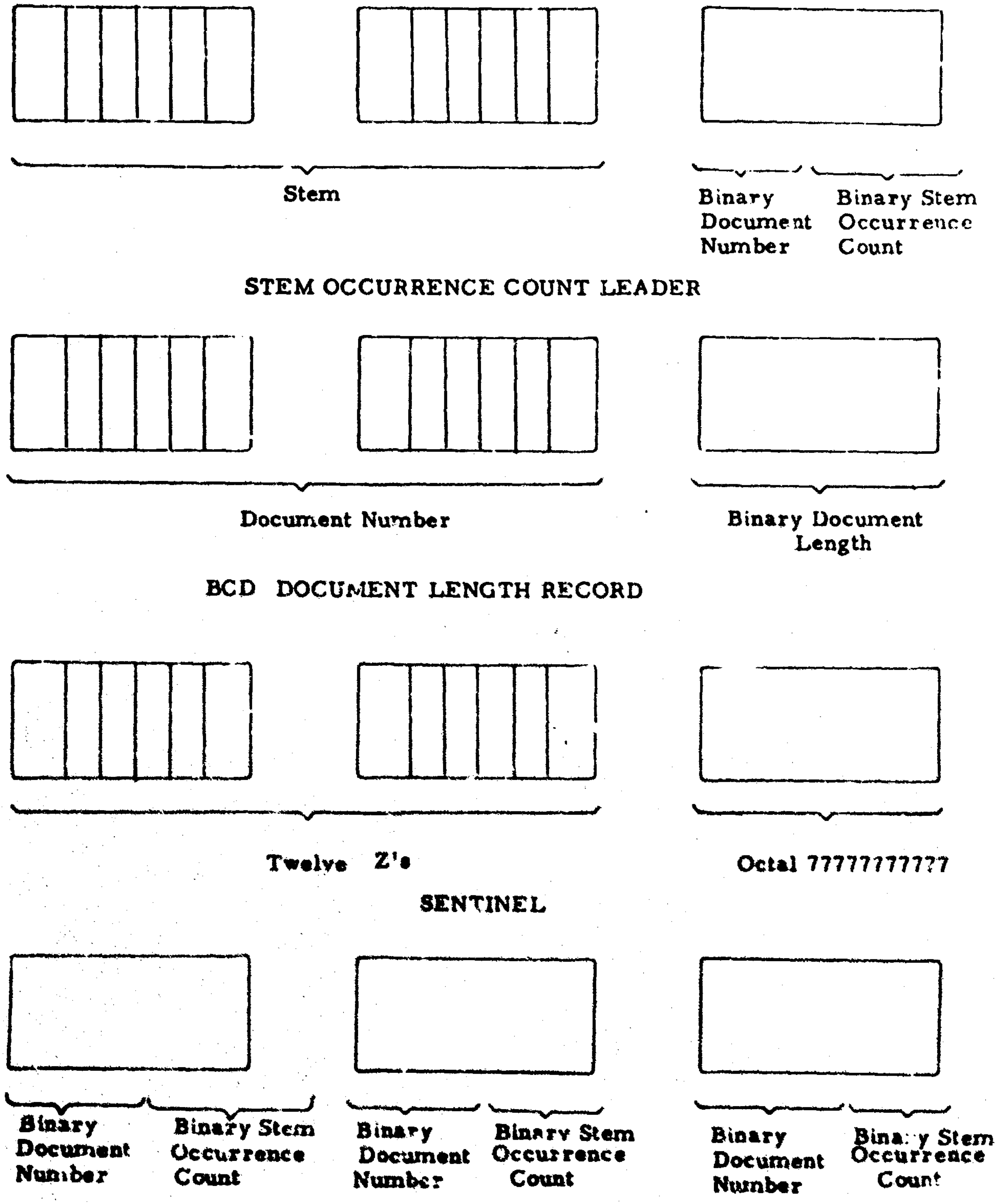
VI. 2 Concept Identification

Concepts will be identified by a two-step process: statistical filtering to remove noncontent stems, and occurrence correlation to identify words of similar occurrence patterns.

The input to the concept identification process will be the weighted concordance of stems and the output will be the content stem-concept dictionary.

VI. 2. 1 Statistical Filter

As discussed in Section III. 2. 1. 1, Dennis' measure, called V_c for convenience in this discussion, is to be used in the selection of content stems for the on-line retrieval system.



STEM OCCURRENCE COUNT LEADER

BCD DOCUMENT LENGTH RECORD

SENTINEL

STEM OCCURRENCE COUNT TRAILER

Figure VI-11. Concordance Physical Record Type

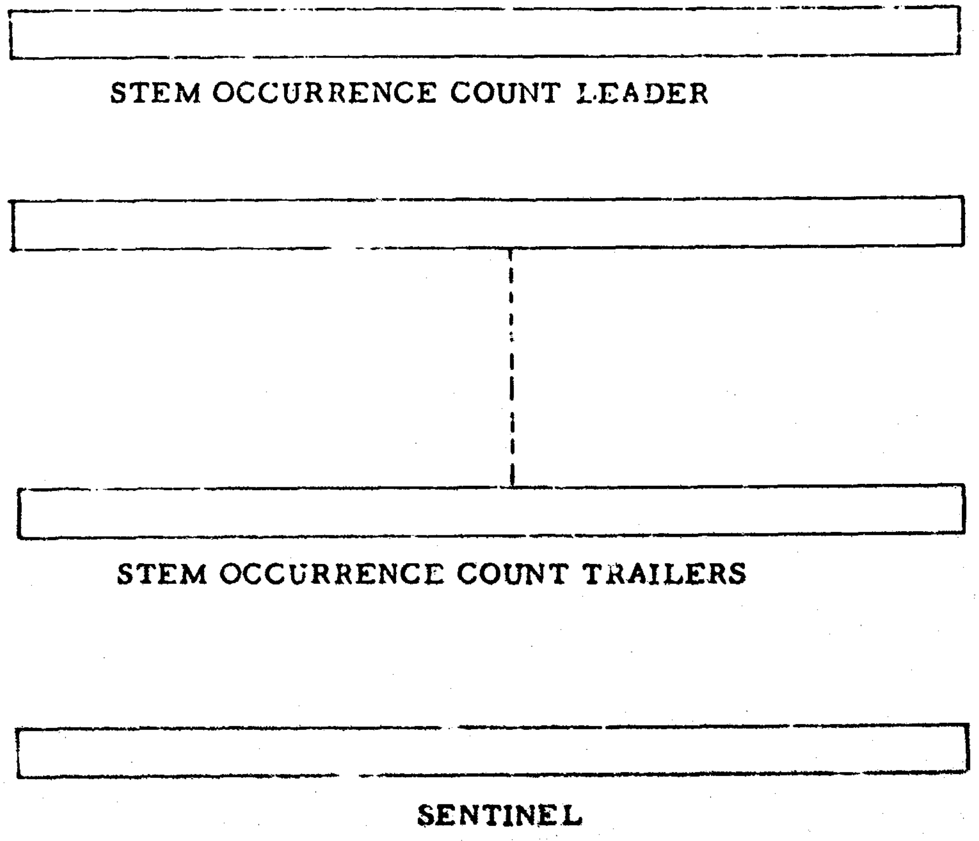


Figure VI-12. Single-Stem Concordance

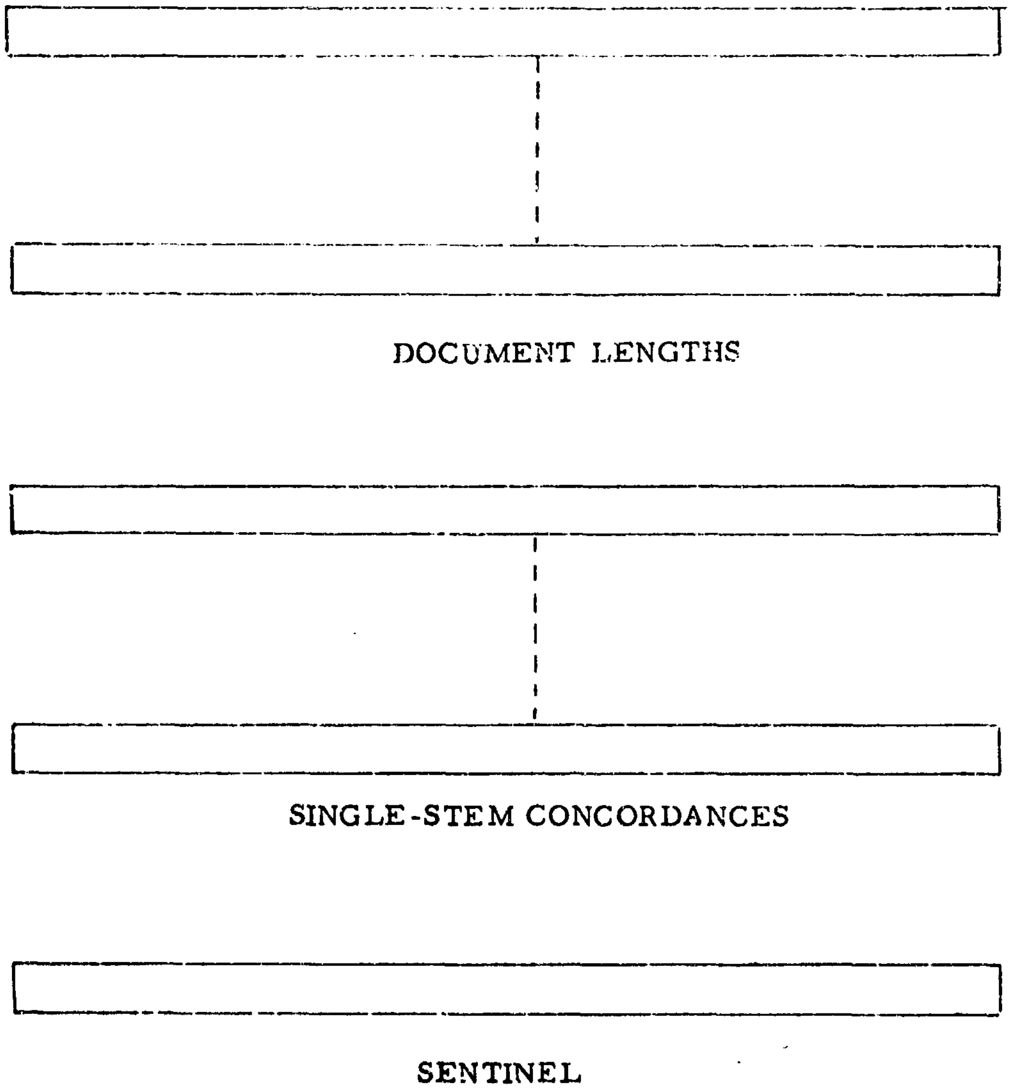


Figure VI-13. Concordance Tape

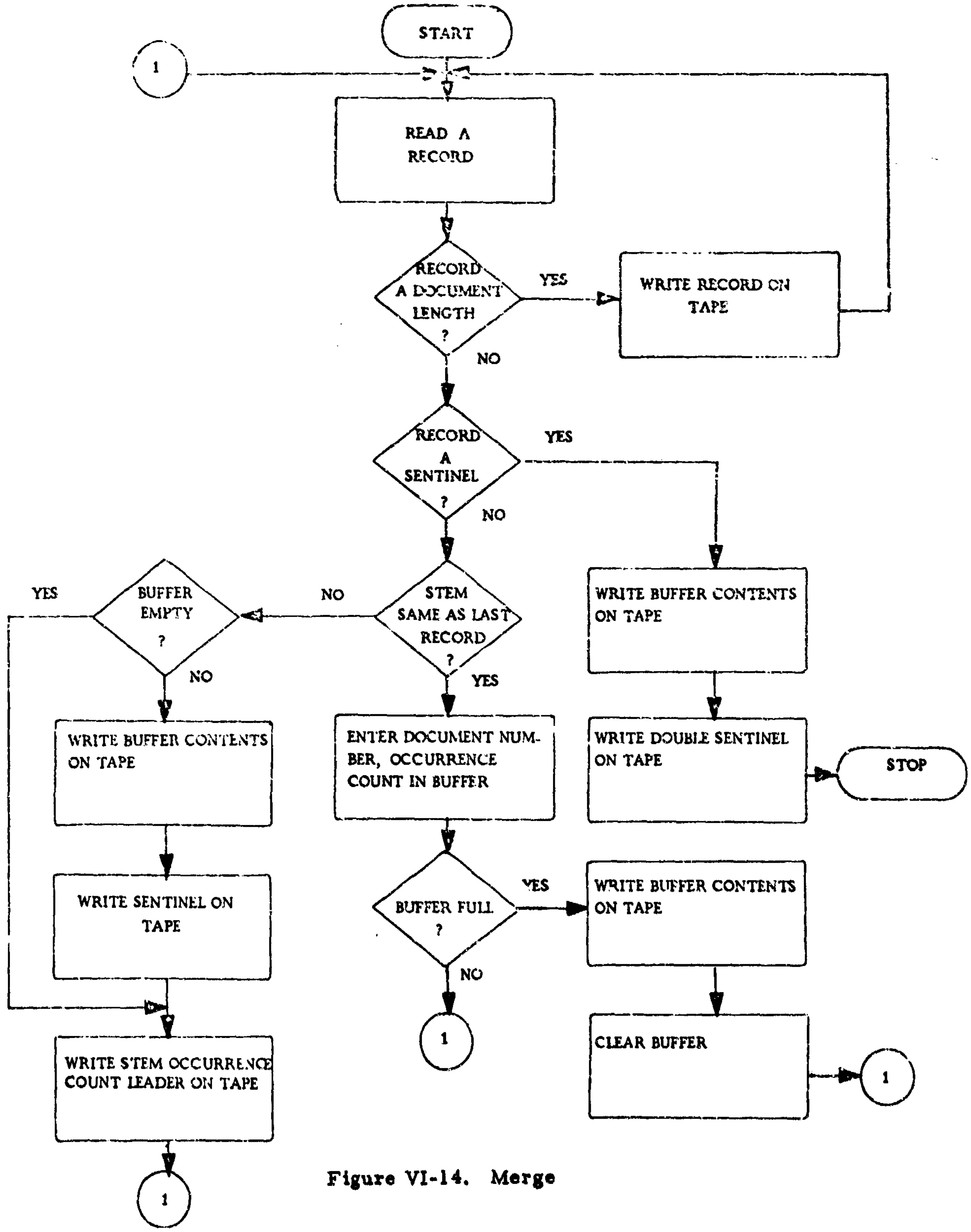

START
1
READ A RECORD
RECORD A DOCUMENT LENGTH ?
YES
WRITE RECORD ON TAPE
NO
RECORD A SENTINEL ?
YES
NO
WRITE BUFFER CONTENTS ON TAPE
WRITE DOUBLE SENTINEL ON TAPE
STOP
YES
BUFFER EMPTY ?
NO
STEM SAME AS LAST RECORD ?
YES
NO
WRITE BUFFER CONTENTS ON TAPE
ENTER DOCUMENT NUMBER, OCCURRENCE COUNT IN BUFFER
WRITE SENTINEL ON TAPE
BUFFER FULL ?
YES
WRITE BUFFER CONTENTS ON TAPE
NO
1
CLEAR BUFFER
1
WRITE STEM OCCURRENCE COUNT LEADER ON TAPE
1

Figure VI-14. Merge

For the present purposes, it is convenient to use a different form from that of Dennis. The computed values are, however, the same as hers.

Let D be the number of documents in the collection, and L_d be the number of stems in document d (that is, the length of document d). The number of occurrences of stem c in document d is $f_{c,d}$, and its normalized form is $g_{c,d} = f_{c,d}/L_d$. The normalized frequency of occurrences over the collection is:

$$\bar{g}_c = \sum_{d=1}^{D} g_{c,d}/D.$$

The number of raw occurrences of stem c in the collection is σ_c, where

$$\sigma_c = \sum_{d=1}^{D} f_{c,d} \cdot$$

Letting $s_c^2 = \sum_{d=1}^{D} (g_{c,d} - \bar{g}_c)^2 \; / \; (D-1)$, then Dennis' measure is

given by $NOCC/EK_c = \sigma_c s_c^2 / \bar{g}_c^2$, and is equal to the present V_c.

In order that V_c can be computed in one pass through the vector $[f_{c,1} \; f_{c,2} \cdots f_{c,D}]$, some algebraic manipulations are necessary. First, let

$$\alpha_c = \sum_{d=1}^{D} \left(\frac{f_{c,d}}{L_d} \right)^2 \text{ and } \beta_c = \sum_{d=1}^{D} \frac{f_{c,d}}{L_d} \quad . \text{ Thus, in}$$

one pass through the vector, α_c, β_c, and σ_c may be computed. Substituting

for s_c^2 and $\bar{g}^2$ in V_c yields

$$V_c = \frac{\sigma_c \sum_{d=1}^{D} (g_{c,d} - \bar{g}_c)^2}{(D-1)\bar{g}_c^2} \quad .$$

Now, $(g_{c,d} - \bar{g}_c)^2 = g_{c,d}^2 - g_{c,d}^2 \bar{g}_c + \bar{g}_c^2$.

From the definitions, $\bar{g}_c = \frac{\beta_c}{D}$ and so

$$V_c = \frac{\sigma_c \sum_{d=1}^{D} \left[\left(\frac{f_{c,d}}{L_d} \right)^2 - 2 \frac{f_{c,d}}{L_d} \frac{\beta_c}{D} + \left(\frac{\beta_c}{D} \right)^2 \right]}{(D-1)\ (\beta_c/D)^2} .$$

But $\sum_{d=1}^{D} \left(\frac{f_{c,d}}{L_d} \right) = \alpha_c$, and $\frac{\beta_c}{D}$ is not a function of d. Therefore,

$$V_c = \frac{D^2}{D-1} \cdot \frac{\sigma_c}{\beta_c^2} \left[\alpha_c - 2 \frac{\beta_c}{D} + D \left(\frac{\beta_c}{D} \right)^2 \right] .$$

As $D >> 1$, $D^2/(D-1) \approx D$. Thus

$$V_c = \frac{D\sigma_c}{\beta_c^2} \left[\alpha_c - \beta_c^2/D \right]$$

$$V_c = \alpha_c \sigma_c D/\beta_c^2 - \sigma_c$$

$$V_c = \sigma_c (\alpha_c D/\beta_c^2 - 1).$$

In computation, recall that the concordance tape contains a table of values of L_d. This is kept in core as each vector of $f_{c,d}$ values is read in. Then only one pass through the values can result in calculation σ_c, α_c, and β_c, as shown in the flowchart.

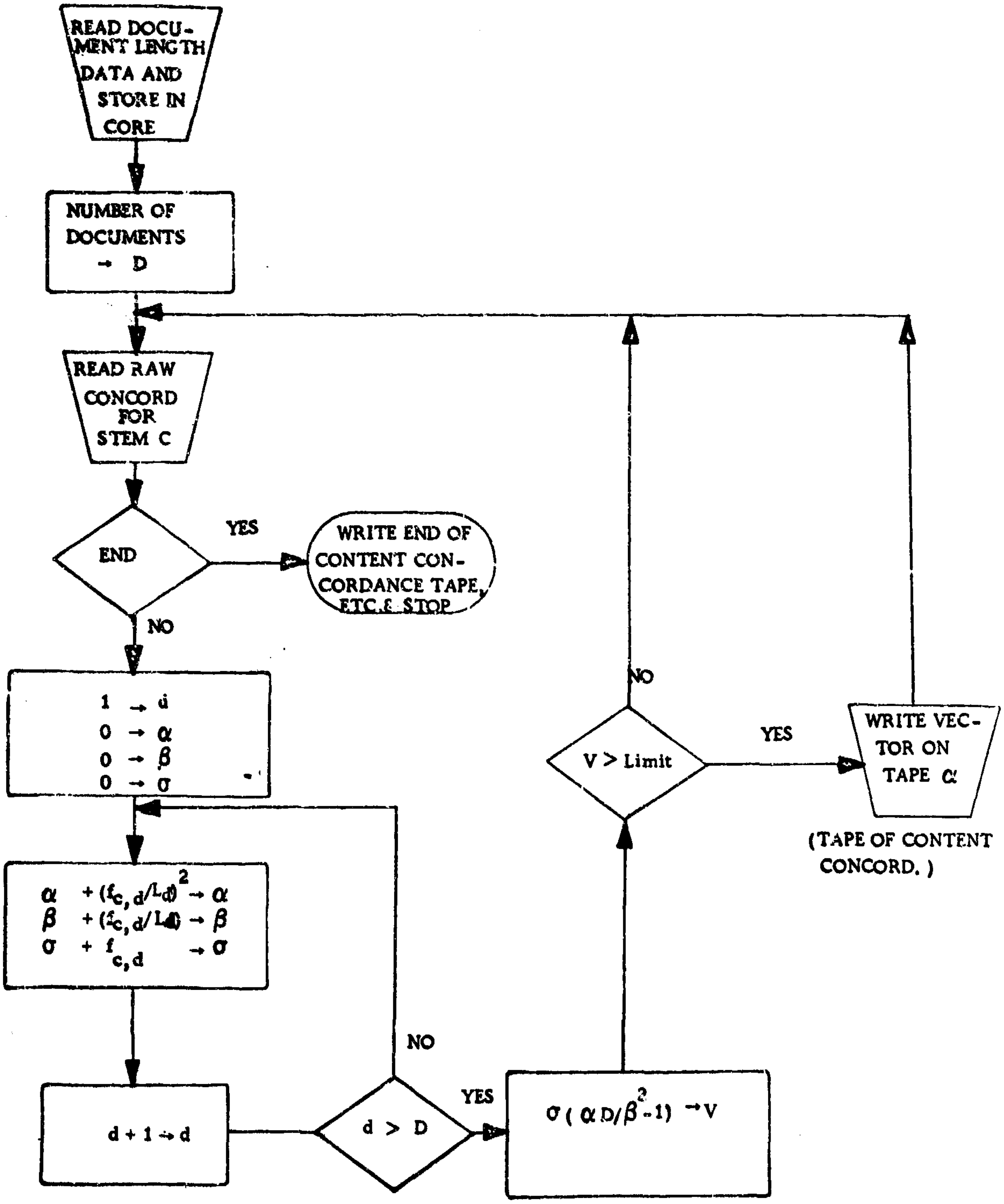

READ DOCUMENT LENGTH DATA AND STORE IN CORE
NUMBER OF DOCUMENTS → D
READ RAW CONCORD FOR STEM C
END
YES
WRITE END OF CONTENT CONCORDANCE TAPE, ETC.& STOP
NO
1 → d
0 → α
0 → β
0 → σ
$\alpha + (f_{c,d}/L_d)^2 \rightarrow \alpha$
$\beta + (f_{c,d}/L_d) \rightarrow \beta$
$\sigma + f_{c,d} \rightarrow \sigma$
d + 1 → d
d > D
NO
YES
$\sigma(\alpha D/\beta^2 - 1) \rightarrow V$
V > Limit
NO
YES
WRITE VECTOR ON TAPE α
(TAPE OF CONTENT CONCORD.)

Figure VI-15. Use of V_c (NOCC/EK$_c$) to Generate Weighted Concordance of Content Stems

Thus, V_c may be computed for each stem in a single pass through the raw concordance. This is evident when one inspects the flowchart of Figure VI-15.

Following the generation of a V_c for each stem, noncontent stems can be removed. Stem c is considered a content stem if V_c exceeds some limit. That limit must be established, and it is also necessary that the working of the algorithm be verified. Indeed, the number 5000 is not sacred and should be justified.

The best tool for this is a table of triplets consisting of the raw stem c, the rank of V_c and the value of V_c. This table may be generated relatively simply by modifying the statistical filter package to write on tape for every stem the alphabetic stem itself and the associated V_c. Standard sort routines are used to sort this tape on V_c, and the tape is then put through a short listing program that adds the rank numbers (1 for the first entry on the sorted tape, etc.). Figuring four triplets per line, the data for 200 items can be printed on one page. In the event that the number of pages becomes unwieldy, it is remembered that a printing abort would cut off those stems with a very low V_c: misspellings, infrequently used proper names and the like.

This printout will have three uses: determination of the cutoff value for V_c, determination of the number of content stems, and debugging the statistical filter routines.

VI. 2. 2 Dimension Reduction and Stem Removal

Figure VI-16 shows the flowchart for this process. The arrays D and W hold document numbers and weights, while the length of the reduced vector is L_{max}.

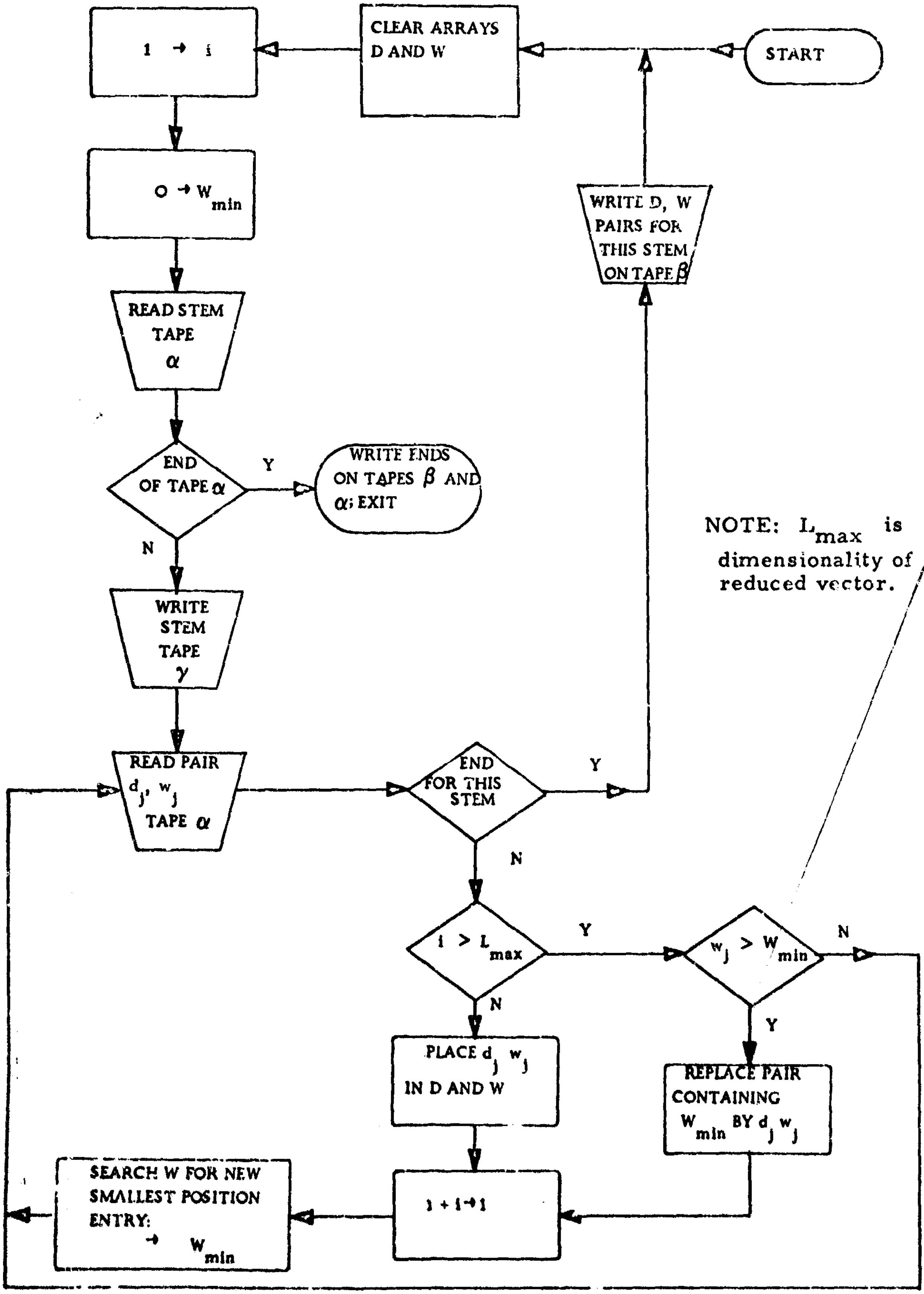

START
CLEAR ARRAYS D AND W
1 → i
O → W_min
READ STEM TAPE α
END OF TAPE α
Y
WRITE ENDS ON TAPES β AND α; EXIT
N
WRITE STEM TAPE γ
READ PAIR d_j, w_j TAPE α
END FOR THIS STEM
Y
WRITE D, W PAIRS FOR THIS STEM ON TAPE β
N
i > L_max
Y
w_j > W_min
N
Y
REPLACE PAIR CONTAINING W_min BY d_j w_j
N
PLACE d_j w_j IN D AND W
i + 1 → i
SEARCH W FOR NEW SMALLEST POSITION ENTRY: → W_min
NOTE: L_max is dimensionality of reduced vector.

Figure VI-16. Dimension Reduction and Stem Removal

VI. 2. 3 Occurrence Correlation and Dictionary Generation

As described in Section III. 2. 1. 3, occurrence correlation is performed in two phases. Phase I identifies certain stems as concept centers; Phase II uses the results of Phase I and the concordance to form the stem-concept dictionary.

VI. 2. 3. 1 Statistical Processing Phase I. Figure VI-17 shows this program, which in turn calls on the Correlation Table Subroutine TGEN. When TGEN is initially called, c_{max} is set equal to two. If additional triplets are required, c_{max} is used to specify a cutoff in order to eliminate triplets already generated. Since cosines do not exceed unity, the initial value of two assures that the first call will locate the largest cosines.

LF is an output parameter of the subroutine, the number of triplets in the table. As the table of triplets is processed, a table of distinct stems (DS) is built. These are candidates for key stems, and during processing of the triplet table stems may be added to or deleted from the distinct stem table. The number of distinct stems in the distinct stem table at any time is S; the last position occupied in the table is indexed by PTL. When an entry is excluded from the table, its vector in core is flagged, in order that additional calls to TGEN do not waste time correlating a stem which cannot be a key stem.

Consider two stems, X and Y, that correlate strongly. During processing of the triplet table to build the distinct stem table:

if X and Y are both in DS, delete Y, flag Y's vector;
if neither X nor Y are in DS, add X, flag Y's vector;
if one is present and one is not, do nothing.

Recall that the triplets are considered in order of decreasing cosine. Therefore, the most strongly clustered cases are treated first. Processing is suspended when S is found to lie in the optimum range.

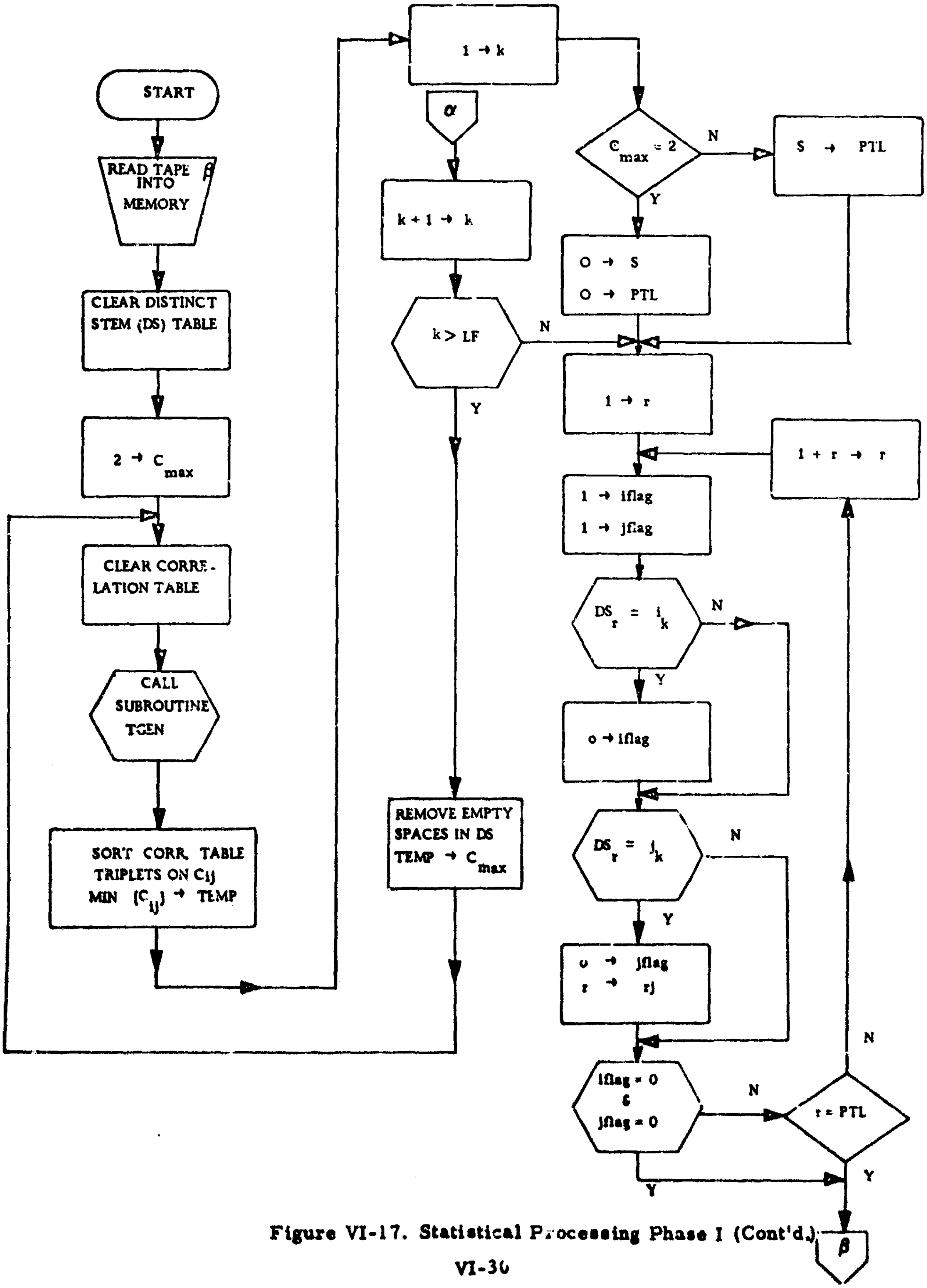


Figure VI-17. Statistical Processing Phase I (Cont'd.)

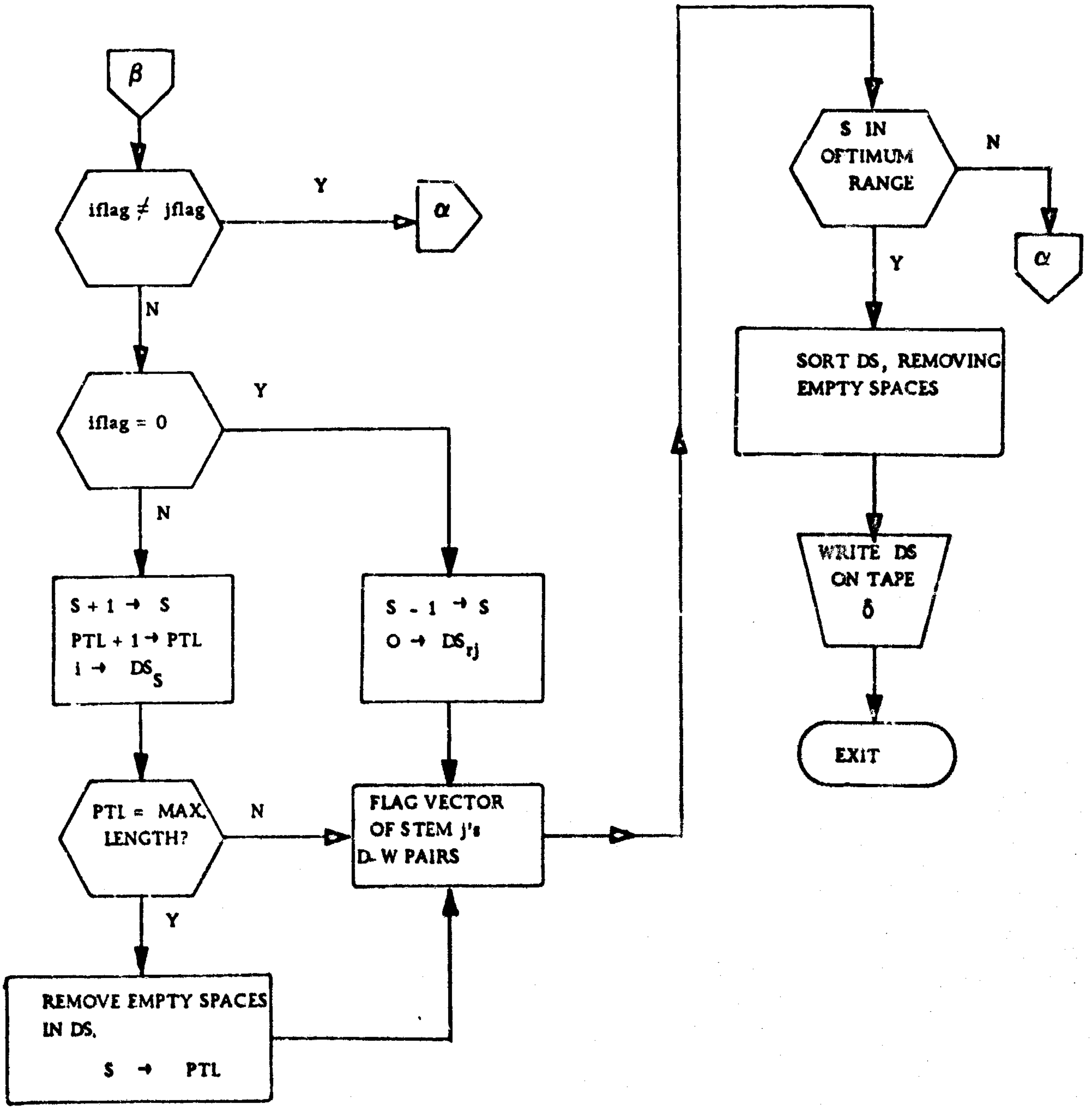


Figure VI-17. Statistical Processing Phase I (Concluded)

Figure VI-18 shows subroutine TGEN, which builds the triplet tables. The parameters c_{max} and LF have been discussed above, as has been the flagging of vectors for stems eliminated from consideration.

VI. 2. 3. 2 Statistical Processing Phase II. This program, illustrated in Figure VI-19, reads key (or concept) stem numbers produced in Phase I and reduced vectors in order to store the vectors for the 1500 key stems in core. The key stems are assigned numeric designations--the concept numbers. Then all content stems (alphanumeric) and their reduced vectors are read, one at a time. They are correlated with the key stem vectors, and the dictionary entries are generated. Dictionary entries consist of a fixed number of concept number-concept weight pairs for each stem. The number of pairs is called M in Figure VI-19; for each pair j the concept number is stored in K_j and the weight in W_j. Before a dictionary entry is written, its elements are sorted on the concept numbers, so that the smallest number appears first.

VI. 2. 3. 3 Aids for Debugging and Performance Evaluation. Several aids are needed in order to get the programs running and to determine what they can do. These are essentially "test points" or "peep holes".

VI. 2. 3. 3. 1 Content-Stem Sequence Number List. This information can be obtained from tape γ, and simply gives a pair for each content stem: (stem sequence number, alphabetic stem). Because the stem sequence numbers are assigned in lexicographic order of the stems, the inverse of the list is not required.

A very short and simple program will read the tape and format and print one page at a time. A page can contain 300 pairs, so the entire vocabulary can be contained on 17 pages. A page consists of six columns of pairs, twenty characters per pair and fifty pairs per column. The ordering scheme is arranged as shown below for convenience:

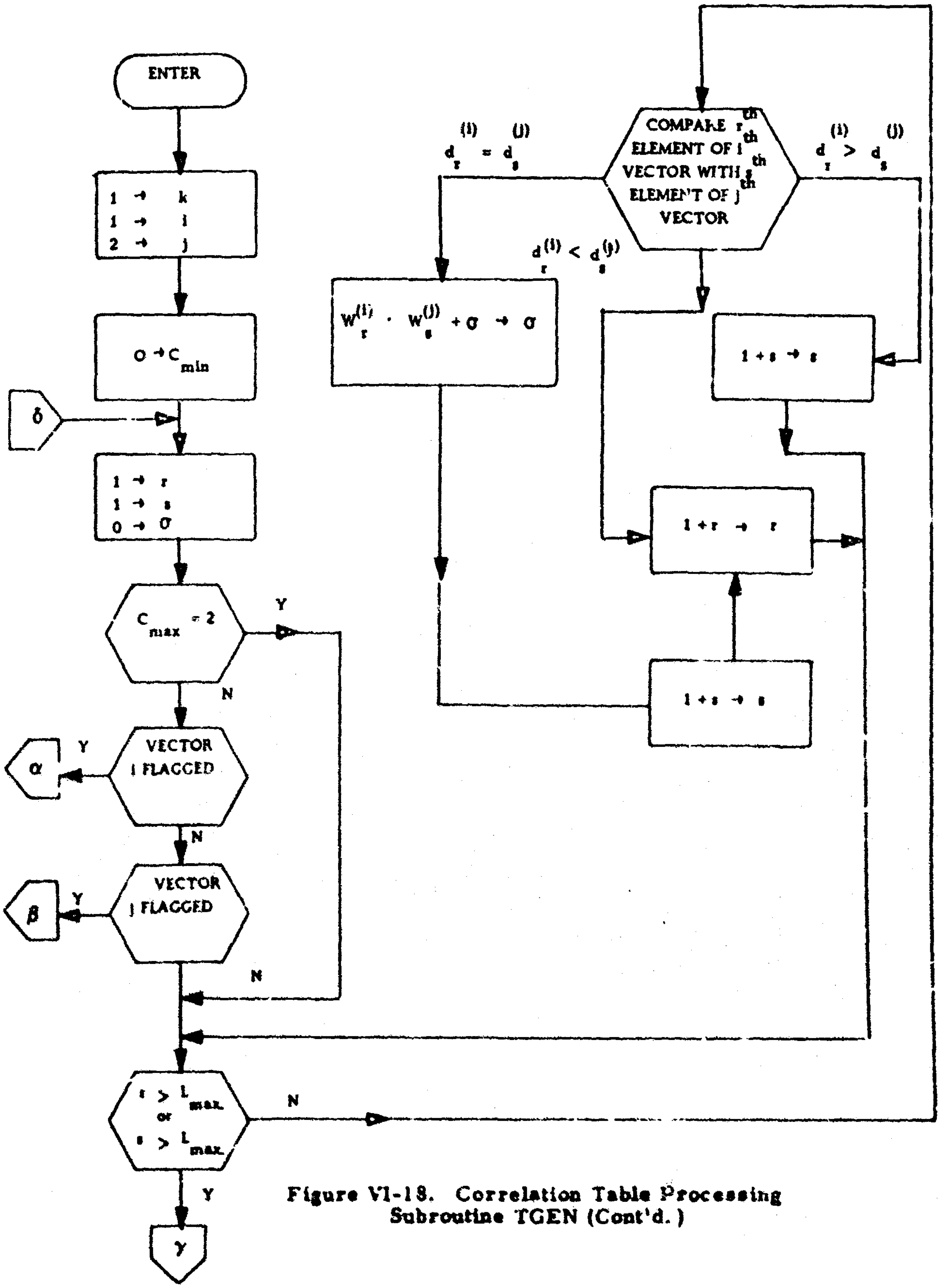


Figure VI-18. Correlation Table Processing Subroutine TGEN (Cont'd.)

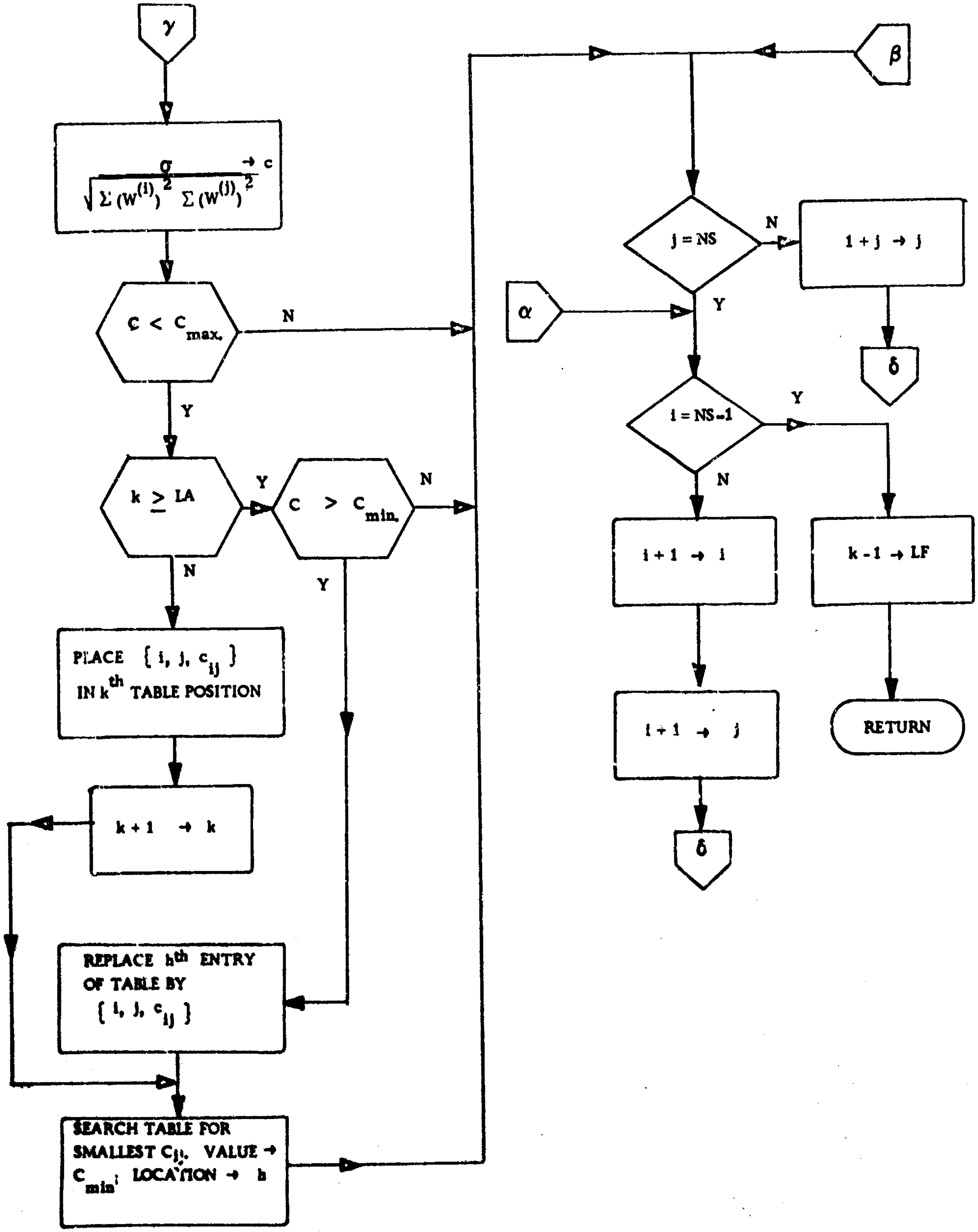

γ
σ / √(Σ(W^(i))² Σ(W^(j))²) → c
c < C max.
N
Y
k ≥ LA
Y
c > C min.
N
Y
N
PLACE {i, j, c ij} IN kth TABLE POSITION
k + 1 → k
REPLACE hth ENTRY OF TABLE BY {i, j, c ij}
SEARCH TABLE FOR SMALLEST C ij. VALUE → C min; LOCATION → h
β
j = NS
N
1 + j → j
δ
α
Y
i = NS-1
Y
N
i + 1 → i
i + 1 → j
δ
k - 1 → LF
RETURN

Figure VI-19. Correlation Table Statistical Processing - Subroutine TGEN (Concluded)

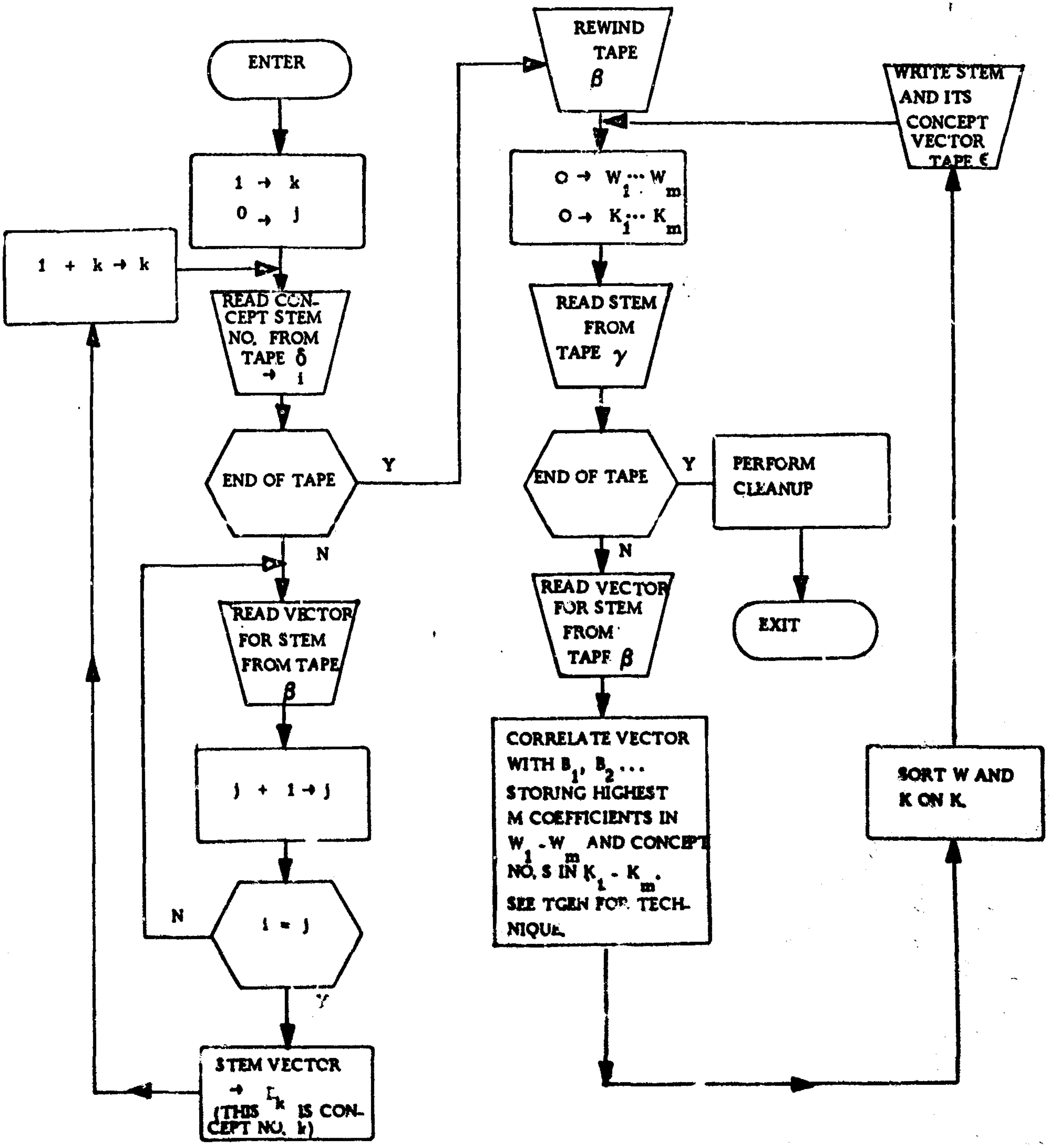

ENTER
1 → k
0 → j
1 + k → k
READ CON-CEPT STEM NO. FROM TAPE δ → i
END OF TAPE
Y
N
READ VECTOR FOR STEM FROM TAPE β
j + 1 → j
i = j
N
Y
STEM VECTOR → L_k (THIS IS CON-CEPT NO. k)
REWIND TAPE β
O → $W_1 \cdots W_m$
O → $K_1 \cdots K_m$
READ STEM FROM TAPE γ
END OF TAPE
Y
N
PERFORM CLEANUP
EXIT
READ VECTOR FOR STEM FROM TAPE β
CORRELATE VECTOR WITH B_1, B_2 ... STORING HIGHEST M COEFFICIENTS IN W_1 - W_m AND CONCEPT NO. S IN K_1 - K_m. SEE TGEN FOR TECH-NIQUE.
SORT W AND K ON K.
WRITE STEM AND ITS CONCEPT VECTOR TAPE ε

Figure VI-19. Statistical Processing - Phase II

1	51	101	151	201	251
2	52	102	152	202	252
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
50	100	150	200	250	300.

This list is to be used in certain debugging operations, and to verify the correctness of tape γ.

VI. 2. 3. 3. 2 <u>Concordance of Stems with Reduced Dimensionality.</u> This concordance exists nowhere in the system, and a complete listing of it would be useless and occupy 500 pages. All that is needed here is enough information to spot check the data-- this may be obtained from tapes β and γ.

VI. 2. 3. 3 <u>Triplet Table.</u> This information is useful in debugging and gaining insight into the workings of the system. It may be obtained simply by adding a few lines of code after the call to TGEN in Phase I (see Figure VI-17). Printing six triplets per line, about twelve pages would be required.

VI. 2. 3. 4 <u>Clustering.</u> For each of the 1500 cluster centers, the history of that center's development is useful both for debugging and for obtaining a "feeling" for the operation of the system in order to adjust the parameters of the statistical filter and the acceptable limits on S. Because this information is critical and should be reviewed by the users and builders of the system, alphabetic stems should be used explicitly, rather than stem sequence numbers. Even though perhaps 30 pages of information may be generated, it is felt that all these data should be available for analysis.

Unfortunately, the data are not directly available. In order to capture them, it is necessary to write on an auxiliary tape every (i, j) pair for which the question "iflag $\neq$ jflag" is answered in the negative

(Figure VI-19) in Phase I. This tape obviously contains fewer than 3500 pairs. A short program is used to read the tape into core and sort the pairs on i. The alphabetic content stems are also read into core. Then for every sequence $(i, j_1), (i, j_2), \ldots, (i, j_m)$ the program simply prints:

$$\mathrm{STEM}_i\text{:}\quad \mathrm{STEM}_{j_1}\quad \mathrm{STEM}_{j_2}\quad \mathrm{STEM}_{j_m}\;.$$

STEM_i is a cluster forming stem, and the other stems have been clustered around it.

VI. 2. 3. 3. 5 Dictionary. Obviously, a hard copy of the dictionary is desired. It is simply obtained from the final dictionary tape and a concept-number alphabetic stem table. The latter may be built in core from tapes γ and ϵ.

A typical dictionary entry would be:

DEJEC: DEPRES 0.761 DESPAIR 0.877 GLOOM 0.468.

VI. 2. 4 Document Concept Vector Generation

The document concept vector generation program reads the microconcordance and produces a tape file of document concept vectors.

The flowchart of Figure VI-20 illustrates document concept vector generation. The stem-concept dictionary described in Section VI. 2. 3 is the "dictionary" referred to in the flowchart. It is used to define the many-many weighted mapping of stems into concept numbers and weights. A dictionary lookup with each stem generates a set of concepts and weights; these are placed in the concept vector under construction. When all the stems in the microconcordance for one document have been read, the concept vector is complete and is written with its accession number onto the concept vector (tape) file. The next microconcordance is then read. This process continues until the last microconcordance has been read.

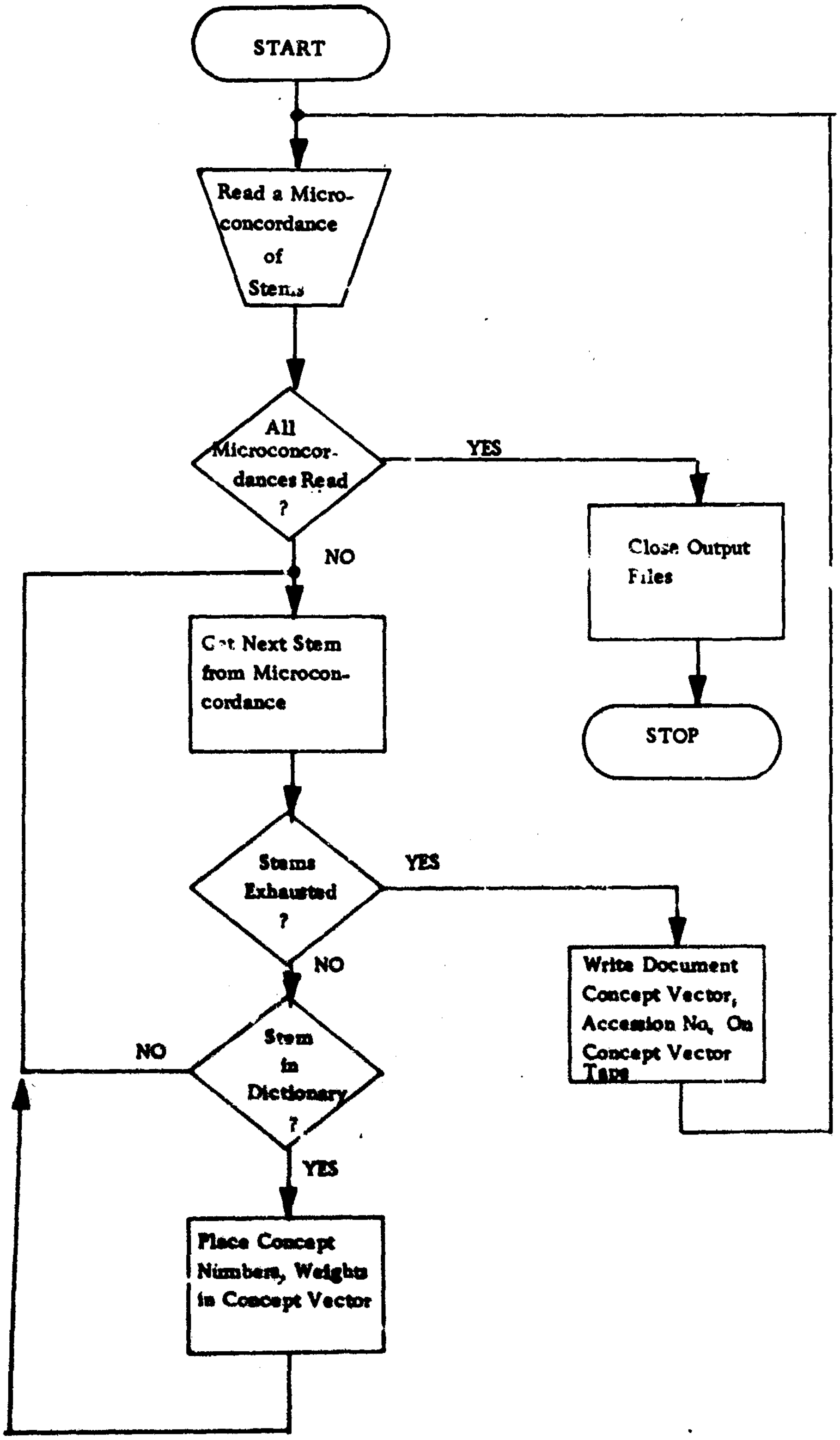

START
Read a Micro-concordance of Stems
All Microconcordances Read ?
YES
NO
Close Output Files
STOP
Get Next Stem from Microconcordance
Stems Exhausted ?
YES
NO
Write Document Concept Vector, Accession No, On Concept Vector Tape
Stem in Dictionary ?
NO
YES
Place Concept Numbers, Weights in Concept Vector

Figure VI-20. Document Concept Vector Generation

Since the stem-concept dictionary includes only content stems, noncontent stems that occur in the microconcordance will be ignored automatically. Thus, no statistical filtering of the microconcordances is required.

VI.3 Loading the On-Line Files

Once the document concept vectors have been generated, the on-line files must be loaded before the On-Line Retrieval system can be used. There are two programs that participate in this process; PILE and SHOVEL. PILE reads the document concept vectors and constructs the concept vector and centroid files on tape. SHOVEL reads the tape files and loads them onto disk, converting them to random-access organization.

VI.3.1 File Construction

PILE constructs only the document concept vector and centroid files; the construction of the document and bibliographic files is a formatting operation that will be performed by another program.*

PILE constructs the document concept vector file first. The clusters within the concept vector file are constructed in reverse order, with the last vector in each cluster written on the file first. Thus, the end-of-cluster indicator will be attached to the first concept vector record written in each cluster. When a record is written, the accession number of the last vector written in the same cluster is used as its forward cluster pointer, and its accession number is stored to be used as the forward cluster pointer for the next concept vector placed in that cluster. When the concept vector file is complete, the pointers remaining in core are available for use as the "beginning of cluster" pointers for the centroid file.

* This program cannot be specified until the exact format of the data base is known.

Figure VI-21 is a flowchart of PILE. Array POINT is used to store the next forward pointer for each of the 1500 clusters. The program begins operation by reading one concept vector and its accession number. A cluster pointer is placed in the output concept vector record for each cluster whose cluster number corresponds to a nonzero weight concept occurrence in the vector. When all the cluster pointers have been placed in the output record, it is written onto the concept vector file, and the next concept vector is read. This process continues until all the concept vectors have been read; at this time the output concept vector file is closed, and construction of the centroid file begins.

The accession numbers remaining in array POINT are used in centroid file construction. Each of these accession numbers is used as the pointer to the first concept vector in its cluster. For example, suppose that POINT (50) contained accession number 346 after concept vector file construction. Then, the 50th centroid record will contain 346 as the pointer to the first concept vector in cluster number 50.

Because each cluster corresponds to one concept, and the concepts are generated by virtue of their occurrence within the document collection, there will not be any empty clusters. If some future change in the data preparation programs should cause empty clusters to occur, however, no change in PILE will be needed. If the j^{th} cluster is empty, POINT(J) will contain a zero after concept vector file construction; this will cause accession number zero to be placed in the j^{th} centroid record as the pointer to the first concept vector in that cluster. As discussed in Chapter V, a zero forward pointer is the end-of-cluster indicator.

VI.3.2 File Loading

SHOVEL transfers the sequential files written by PILE and the document formatting programs to random-access GECOS III permanent files. Whenever the permanent files used to store the on-line files are

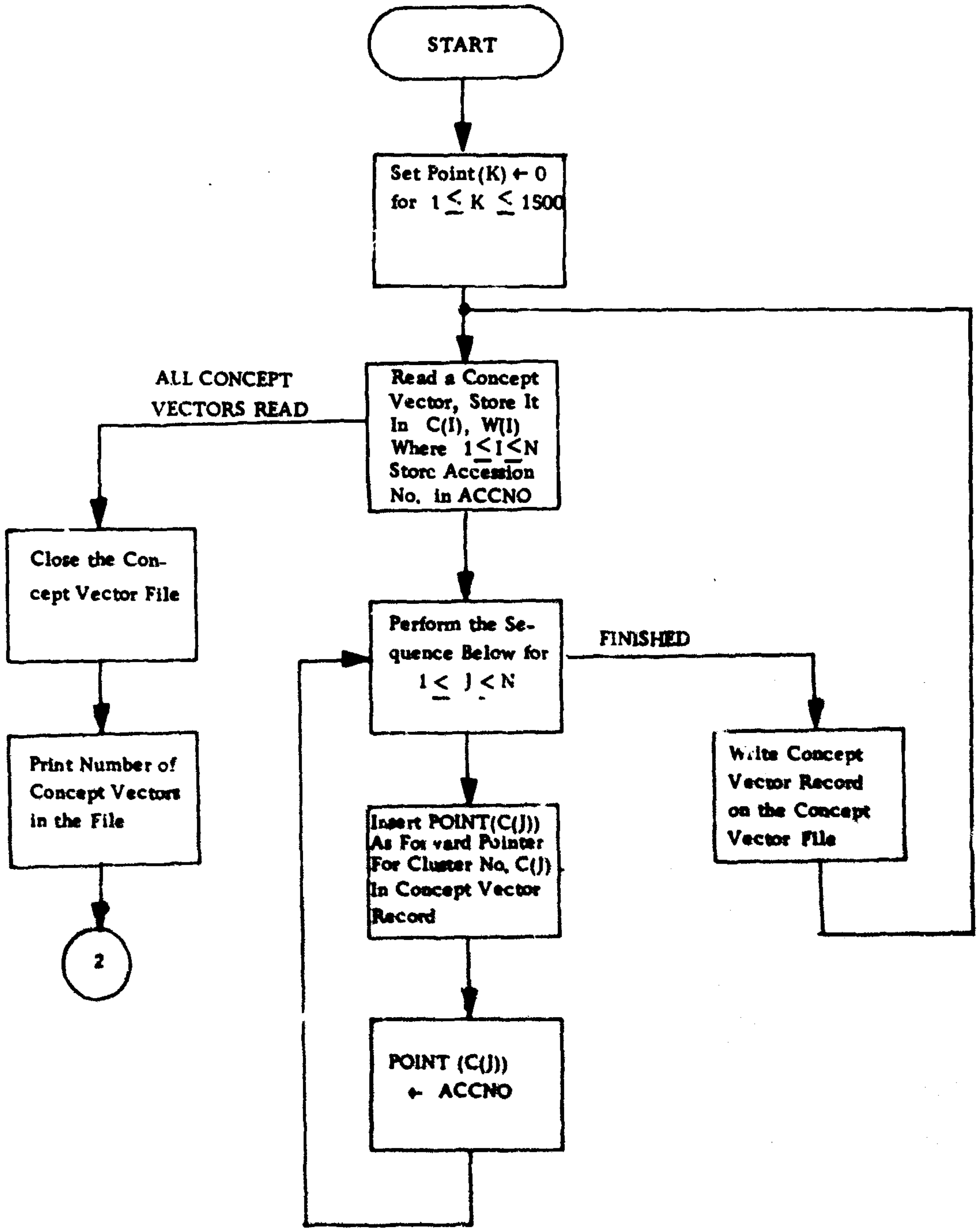
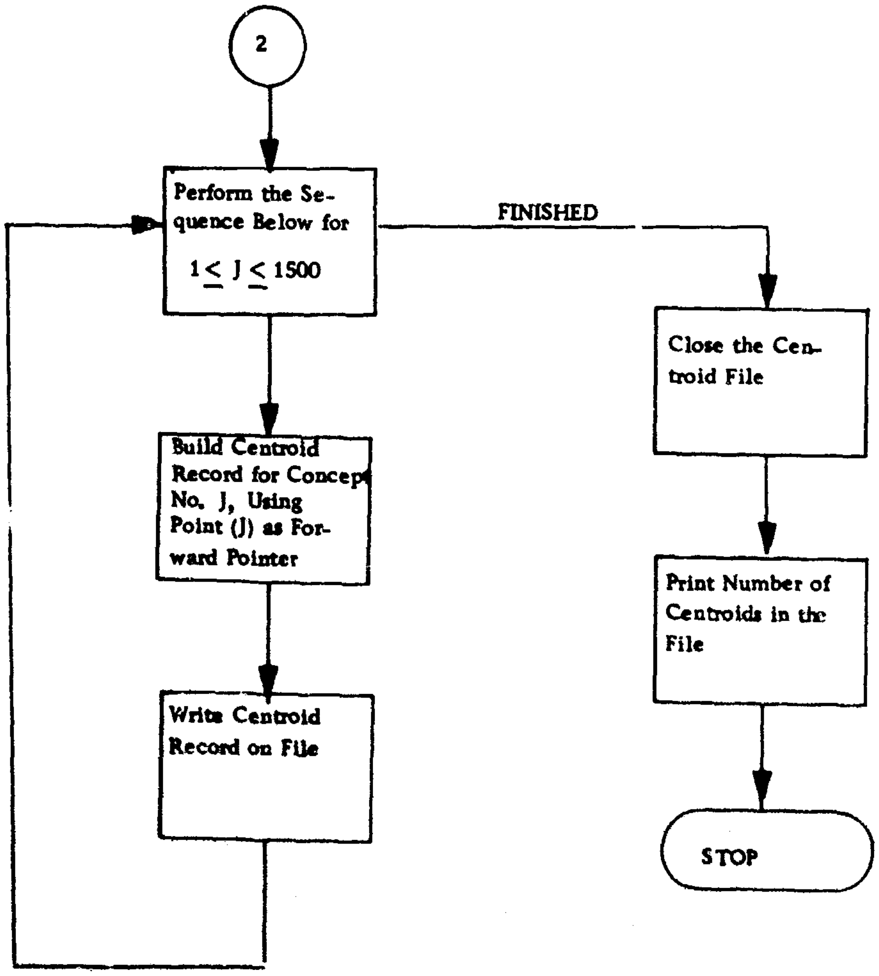

START
Set Point(K) ← 0 for 1 ≤ K ≤ 1500
Read a Concept Vector, Store It In C(I), W(I) Where 1 ≤ I ≤ N Store Accession No. in ACCNO
ALL CONCEPT VECTORS READ
Close the Concept Vector File
Print Number of Concept Vectors in the File
2
Perform the Sequence Below for 1 ≤ J ≤ N
FINISHED
Write Concept Vector Record on the Concept Vector File
Insert POINT(C(J)) As Forward Pointer For Cluster No. C(J) In Concept Vector Record
POINT (C(J)) ← ACCNO

Figure VI-21. PILE (Cont'd.)



2
Perform the Sequence Below for
1 ≤ J ≤ 1500
FINISHED
Build Centroid Record for Concept No. J, Using Point (J) as Forward Pointer
Write Centroid Record on File
Close the Centroid File
Print Number of Centroids in the File
STOP

Figure VI-21. PILE (Concluded)

unloaded, SHOVEL must be run to reload the files before the on-line system can be used. Thus, while PILE will be run only when the data base or automatic indexing structure is to be changed, SHOVEL may be run fairly often. Because of this situation, the complexity of SHOVEL has been reduced as much as possible by assigning all functions that could be performed by either PILE or SHOVEL to PILE.

SHOVEL also constructs COUNTS, the file that indicates the size of all loaded files. This file is written after all the other on-line files are loaded, and is used by FETCH to prevent invalid I/O operations.

Figure VI-22 illustrates SHOVEL.

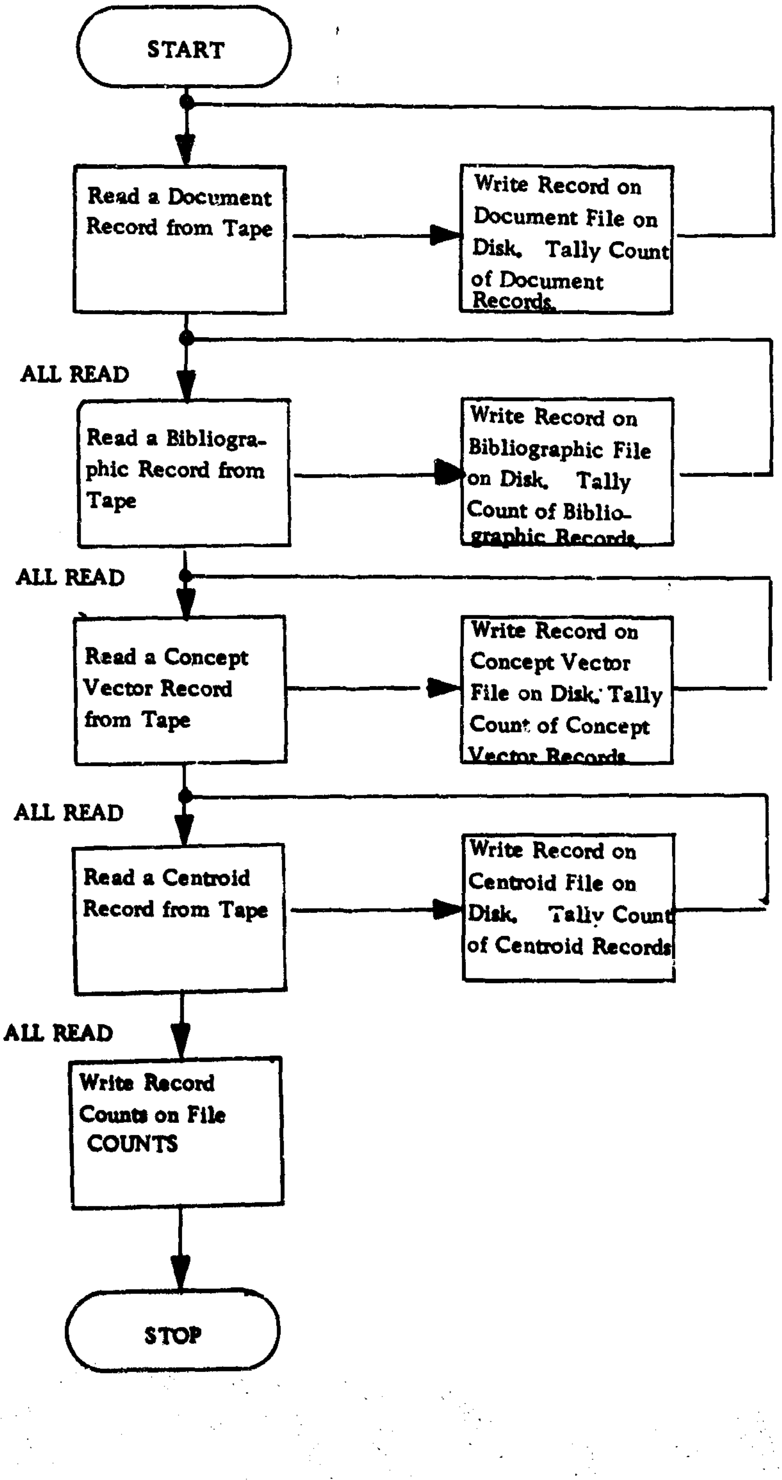

START
Read a Document Record from Tape
Write Record on Document File on Disk. Tally Count of Document Records.
ALL READ
Read a Bibliographic Record from Tape
Write Record on Bibliographic File on Disk. Tally Count of Bibliographic Records.
ALL READ
Read a Concept Vector Record from Tape
Write Record on Concept Vector File on Disk. Tally Count of Concept Vector Records
ALL READ
Read a Centroid Record from Tape
Write Record on Centroid File on Disk. Tally Count of Centroid Records
ALL READ
Write Record Counts on File COUNTS
STOP

Figure VI-22. SHOVEL

SECTION VII

ON-LINE SYSTEM - IMPLEMENTATION DETAILS

This Chapter presents a detailed view of the implementation of the various program modules that comprise the On-Line Retrieval System, elaborating on the introduction presented in Chapter V. To avoid duplication, subprograms that are part of the data preparation programs as well as the On-Line System are described only in Chapter VI; this Chapter references those descriptions.

VII. 1 DIALOGUE

Program module DIALOGUE is described in Chapter IV, and is therefore omitted from this discussion. Note that the dialogue processor controls the sequence of operation of all the program modules discussed in this chapter.

VII. 2 FETCH

FETCH will be written as a subprogram. Given a record number and file number*, FETCH returns the size of the record in words in SIZE and the record in the first SIZE words of RECORD.

To obtain all the document concept vectors in a cluster, FETCH will be called first to obtain the centroid for that cluster. The pointer in the centroid record to the beginning of the concept cluster then gives the accession number of the first concept vector in the cluster. Using that record number, the first document concept vector in a cluster can

* Where a unique integral file number is assigned to the centroid, concept vector, bibliographic, and document files.

be FETCHed. The pointer in the second record FETCHed to the next record in the cluster can be used to move through the cluster. In this way, all concept vectors in any given cluster can be fetched, one at a time.

It is very important that FETCH never try to read from a file that is not loaded or try to read a record number higher than the highest in a file. If either of these is attempted, GETRC will cause the on-line system to be aborted. This can cause considerable inconvenience to the user of the system, because he would lose his current query. To avoid this problem, FETCH checks all file access requests and all record numbers for validity before attempting to read from the on-line files. FETCH obtains information concerning file size and the identity of files that are loaded from another permanent file, COUNTS, which will be written by SHOVEL, as described in Section VI. 3. 2. On the first call to FETCH, FETCH will compare the requested file number and record number with the stored permissible values, without requiring an access to COUNTS for each call to FETCH. Because of this, the user cannot cause the on-line system to abort by causing a FETCH with an invalid accession number.

Figure VII-1 is a flowchart of FETCH. On the first call to FETCH (recognized by a flag setting), permanent file COUNTS is read to obtain the number of records that have been stored on disk in each file. On calls after the first, this read operation will not be repeated.

The file number and record number are checked for validity, and, if they are both acceptable, the requested record is read from the appropriate file and stored in array RECORD. In SIZE is stored the length of the record in words. If the requested file or record number is invalid, -1 or -2, respectively, is returned in SIZE to indicate that a read did not take place and that the contents of RECORD are to be ignored.

VII. 3 CHOOSE

Program module CHOOSE, given a query vector by DIALOGUE,

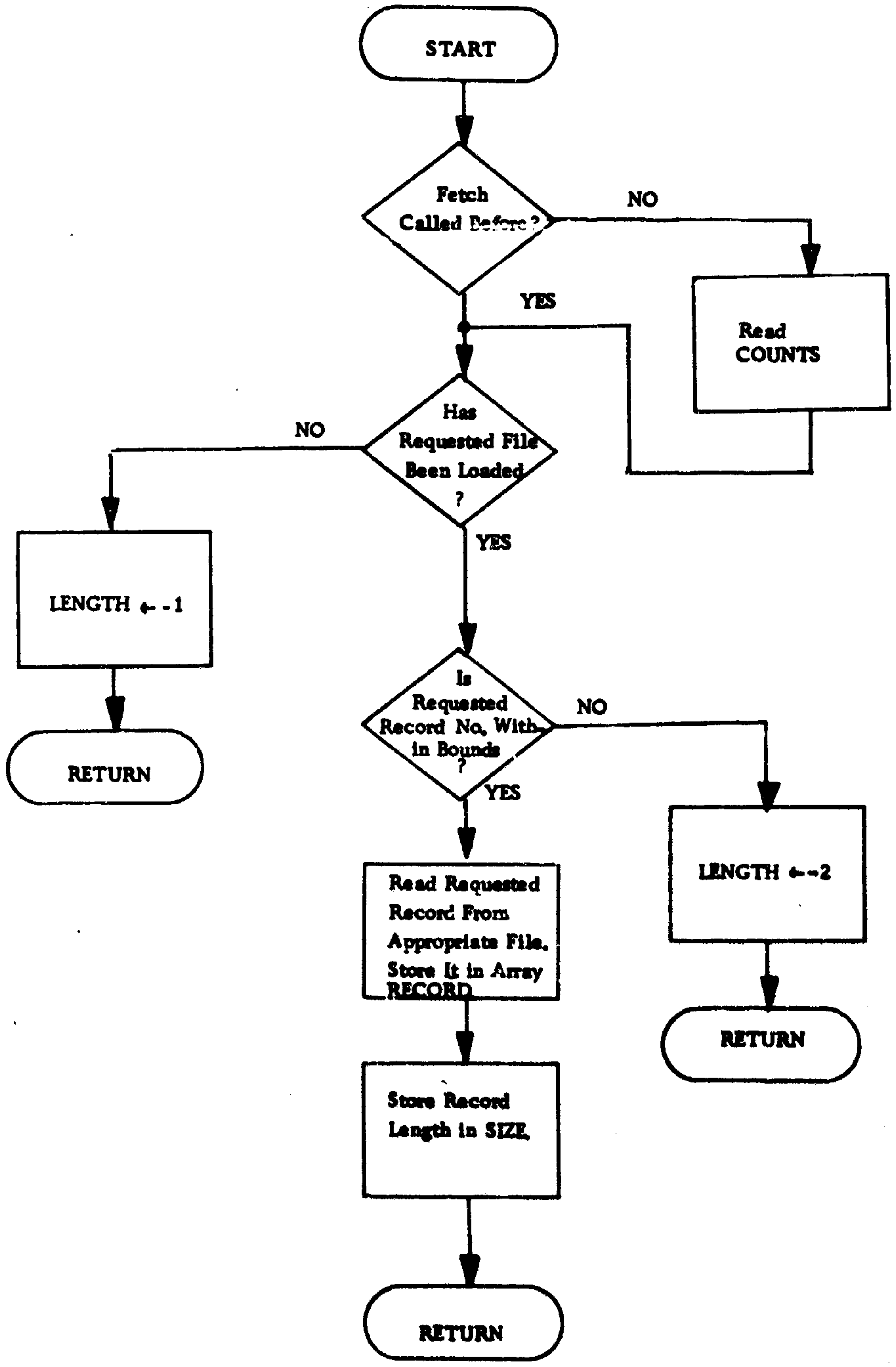

START
Fetch Called Before?
NO
YES
Read COUNTS
Has Requested File Been Loaded ?
NO
YES
LENGTH ← -1
RETURN
Is Requested Record No. Within Bounds ?
NO
YES
LENGTH ← -2
RETURN
Read Requested Record From Appropriate File. Store It in Array RECORD
Store Record Length in SIZE
RETURN

Figure VII-1. FETCH (RECNO, FILENO, SIZE RECORD)

returns to DIALOGUE a ranking table of the accession numbers of the documents whose concept vectors have the highest correlation coefficients with the query vector. CHOOSE calls program module FETCH to obtain centroids and document concept vectors from the data base; therefore, whenever CHOOSE is loaded, FETCH will also be loaded along with it. CHOOSE also calls subprogram CORRELATE, which, given two vectors, computes their correlation coefficient. This routine will use the cosine as the measure of correlation, as discussed in Section III. 2. 3. The correlation measure used for CHOOSE will be identical to the one used in Phases I and II of Statistical Processing (see Section VI. 2. 3); this commonality of algorithms will be exploited.

Figure VII-2 is a flowchart of the operation of CHOOSE. CHOOSE determines which clusters are to be searched by CORRELATing each centroid vector with the query vector. The start-of-cluster pointer (see Section V. 2. 4) of each centroid with a nonzero correlation with the query vector is stored in array GET.

When the entire centroid file has been scanned, selective search of the document concept vector file begins. FETCH is called repeatedly to obtain all the document concept vectors in each cluster whose first entry is pointed to by an element of CET. Each concept vector is compared with a list of concept vectors that have been fetched during this call to CHOOSE, and if it is on the list it is not processed further. This list will be stored as a string of bits, one per document. When a document is CORRELATed, its bit in the list will be set. For efficiency, the list will be maintained and tested by GMAP subroutines.

If the document concept vector has not been processed before, CORRELATE is called to obtain its correlation coefficient with the query vector. If the coefficient is greater than the lowest coefficient in the ranking table of document concept vectors, then it is added to the ranking table, replacing the entry of lowest correlation. Then the next concept vector in the cluster is FETCHed, and the process repeated, until all the vectors in the cluster have been FETCHed. Each cluster pointed to by

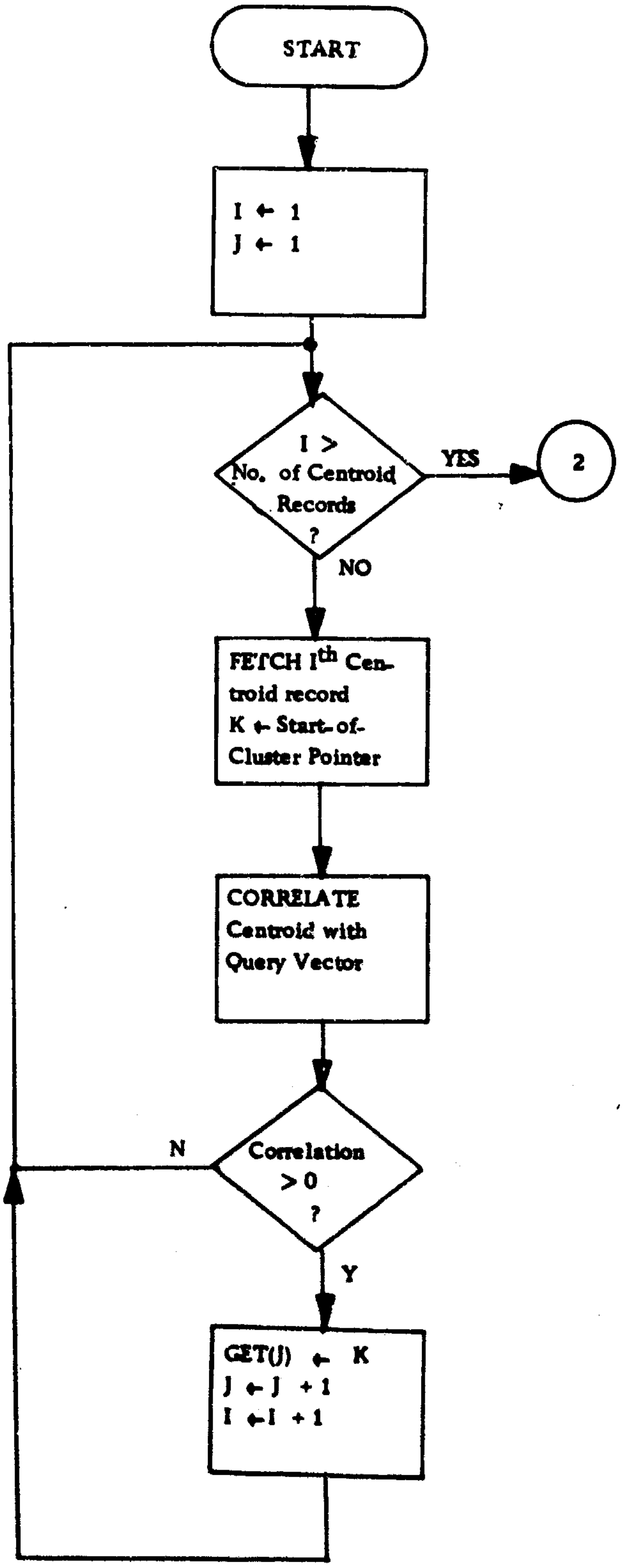


Figure VII-2. CHOOSE (Cont'd.)

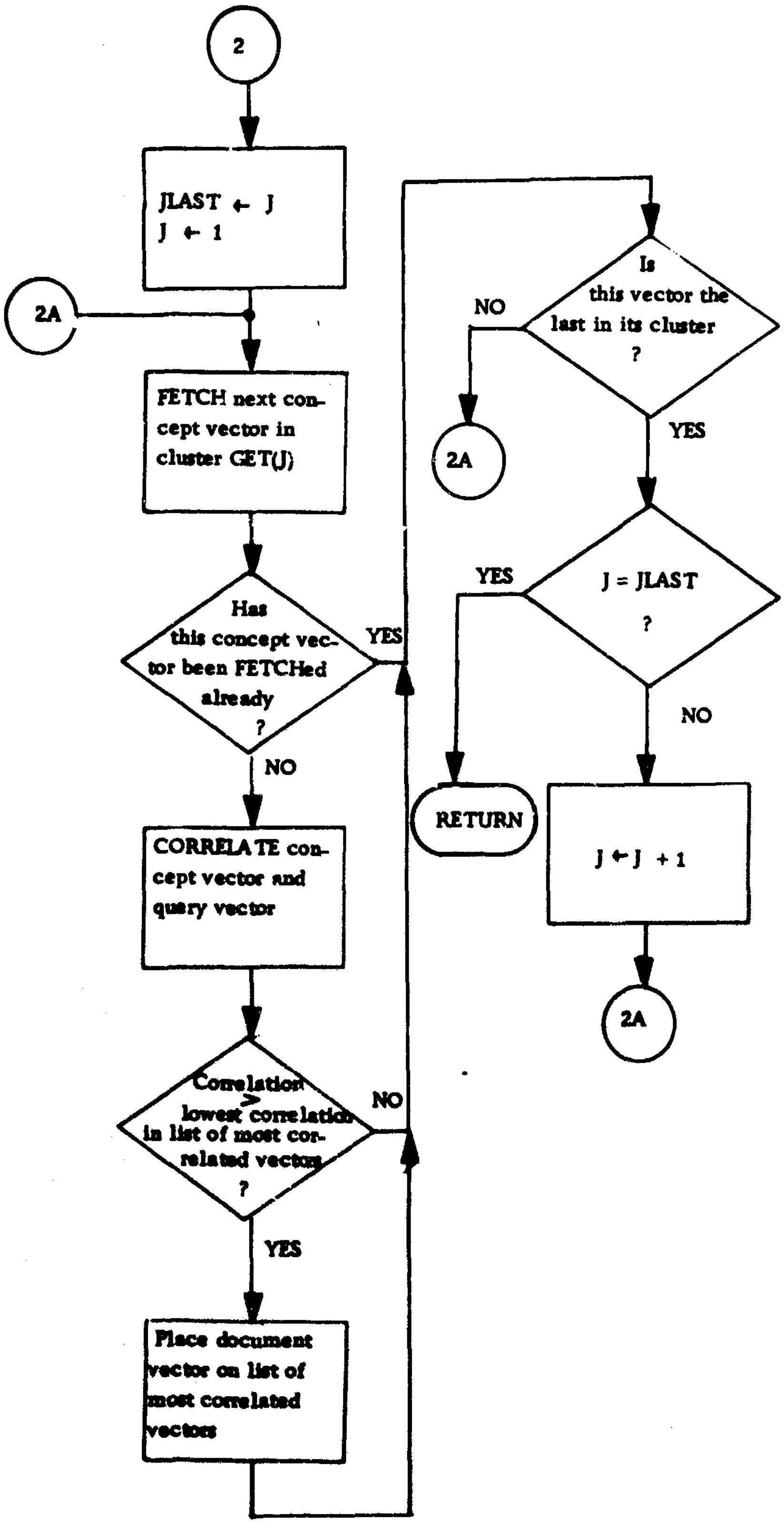

2
JLAST ← J
J ← 1
2A
FETCH next concept vector in cluster GET(J)
Has this concept vector been FETCHed already ?
YES
NO
CORRELATE concept vector and query vector
Correlation > lowest correlation in list of most correlated vectors ?
NO
YES
Place document vector on list of most correlated vectors
Is this vector the last in its cluster ?
NO
2A
YES
J = JLAST ?
YES
NO
RETURN
J ← J + 1
2A

Figure VII-2. CHOOSE)(Concluded)

an element of GET is scanned; when the scanning is complete, control is returned to DIALOGUE.

VII. 4 BUILD

Program module BUILD receives a plain text query from DIALOGUE and builds a query vector.

Figure VII-3 is a flowchart of BUILD. BUILD has three labelled common areas, namely the query vector, list of generated stems, and list of ignored stems, that are used to pass output to DIALOGUE. The list of generated stems is used to contain the stem of each word in the query; the list of ignored stems is used for the stems in the query that are not common words and do not appear in the stem-concept dictionary.

BUILD begins execution by clearing the output areas. One word at a time is GRABbed from the query (for a flowchart of GRAB, see Figure VI-6), and stem analysis is performed by calling STEM (flowcharted in Figure VI-7). If the word was a common word, STEM does not return a stem, and the next word is GRABbed. If a stem was found, then a dictionary lookup on the stem is performed. If the stem is not a content stem, then it will not appear in the dictionary; when this occurs, the stem is placed in the list of ignored stems. If the stem is a content stem, then the lookup will result in a set of concepts and weights that are added to the query vector under construction. This process is repeated with each word in the query until the query is exhausted, at which time control is returned to DIALOGUE.

VII. 5 CHAT

While CHAT is the name of a program module, it is not also the name of a subprogram that is called by DIALOGUE, as are the other program module names. CHAT consists of three subprograms that are all called directly by DIALOGUE: BELCH, GULP, and OUT.

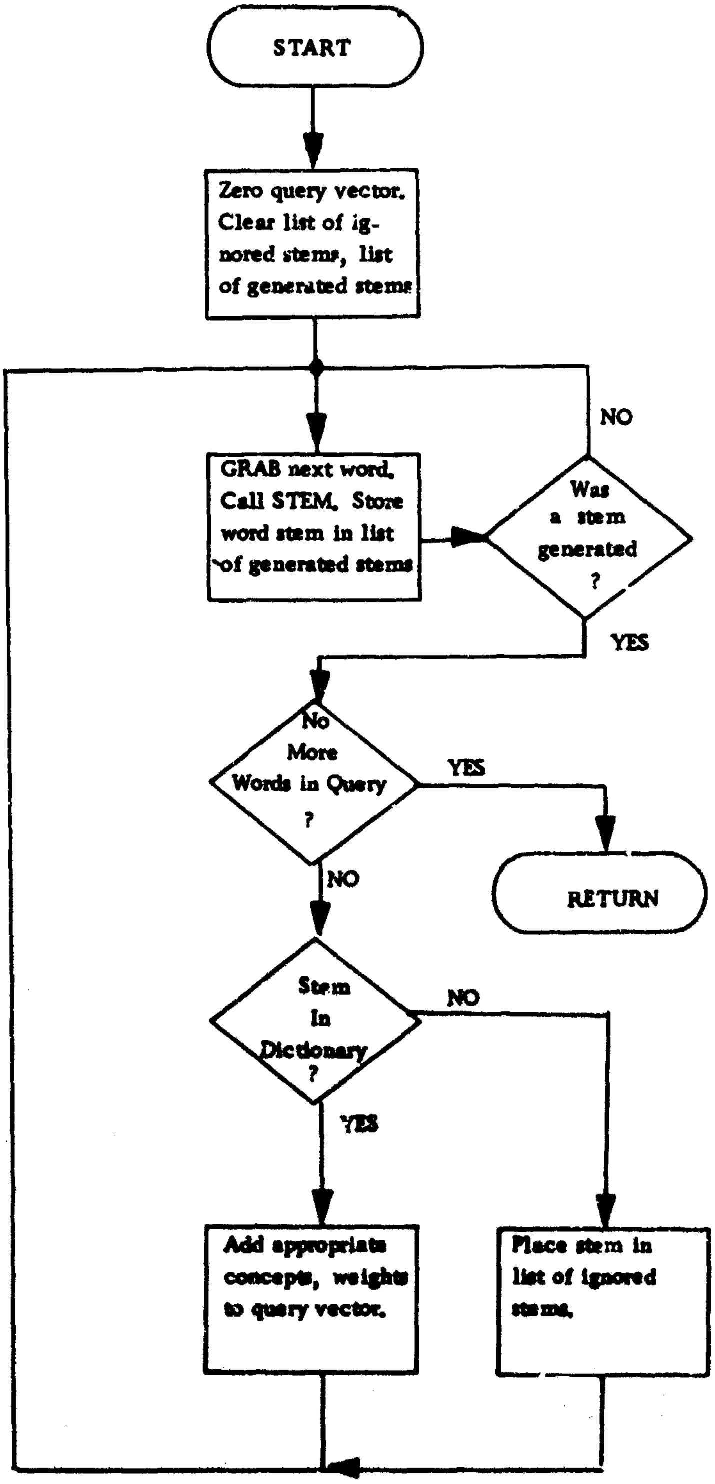

START
Zero query vector. Clear list of ignored stems, list of generated stems
GRAB next word. Call STEM. Store word stem in list of generated stems
Was a stem generated ?
NO
YES
No More Words in Query ?
YES
RETURN
NO
Stem In Dictionary ?
NO
YES
Add appropriate concepts, weights to query vector.
Place stem in list of ignored stems.

Figure VII-3. BUILD

OUT is called to print a standard system message at the remote terminal. To print other data remotely, BELCH is called. GULP is used for all reading from the remote terminal.

BELCH and GULP make use of the direct access capability provided by GERTS to permit a slave program executing in the batch world to communicate with a remote terminal during execution. This is performed by calling GECOS III I/O routines with the Master Mode Entry instruction. In order to use the MME instruction, BELCH and GULP must be written in GMAP assembly language.

VII.5.1 BELCH

Figure VII-4 is a flowchart of BELCH. The squares to the left of the flowchart give the names of various program segments. Labelled common area CUD is used by DIALOGUE to communicate with BELCH, as follows:

Position	Contents
1-12	Data to be printed.
13	Control; tells BELCH what is to be done.
14	Completion status.

If Line (13) = 0, then BELCH will translate the string in LINE (1) to LINE (13) from GE 6-bit code to ASCII 9-bit code, deleting trailing blanks, and transmit the line, without adding any characters. If LINE (13) = -1, the contents of LINE (1) are sent without translation or deletion of trailing blanks. If LINE (13) = 1, BELCH transmits an end-of-line string consisting of carriage return, line feed, rubout, rubout.

Various termination statuses produce different actions from BELCH. If the terminal has disconnected, BELCH terminates this run of

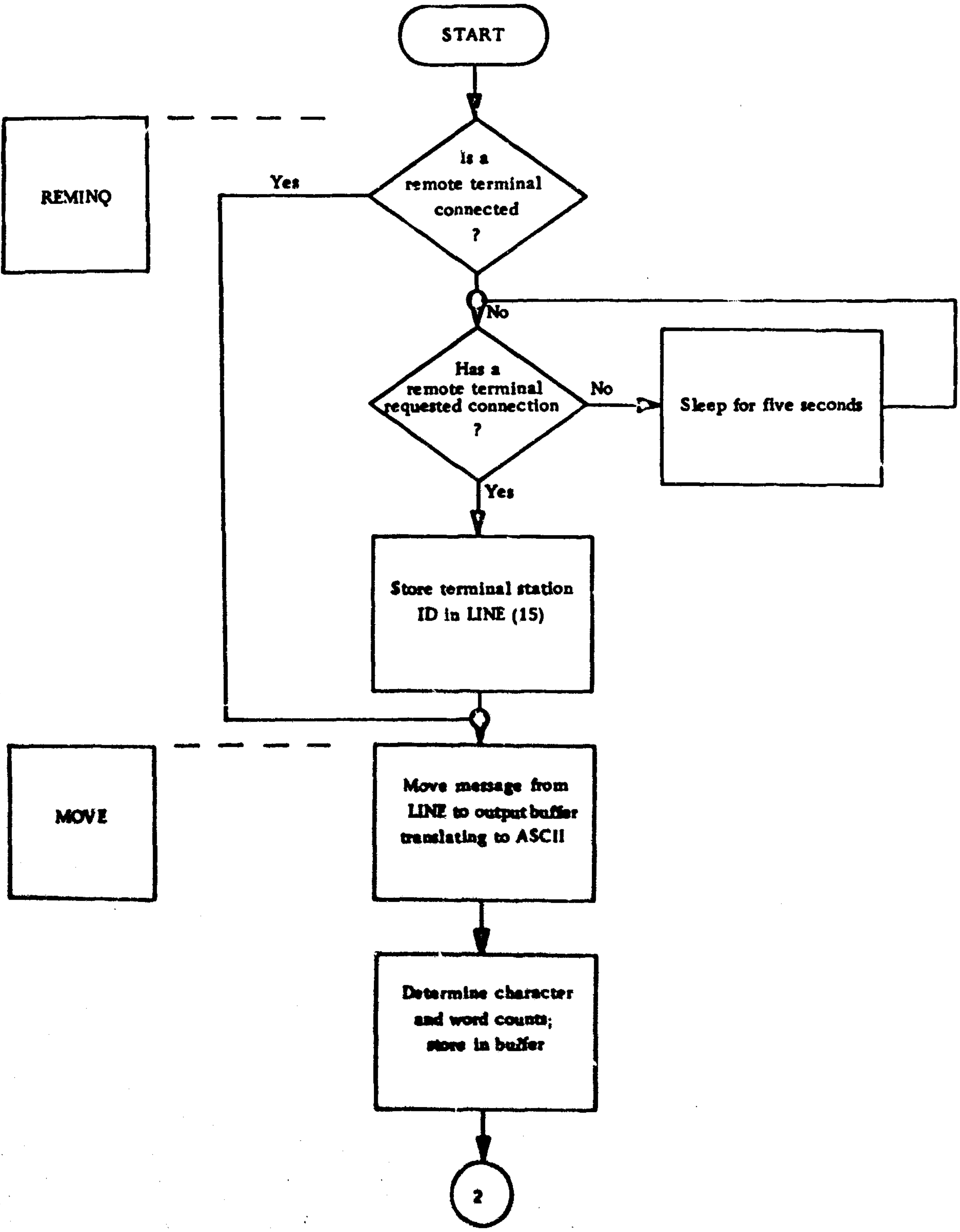

START
REMINQ
Is a remote terminal connected ?
Yes
No
Has a remote terminal requested connection ?
No
Sleep for five seconds
Yes
Store terminal station ID in LINE (15)
MOVE
Move message from LINE to output buffer translating to ASCII
Determine character and word counts; store in buffer
2

Figure VII-4. BELCH (Cont'd.)

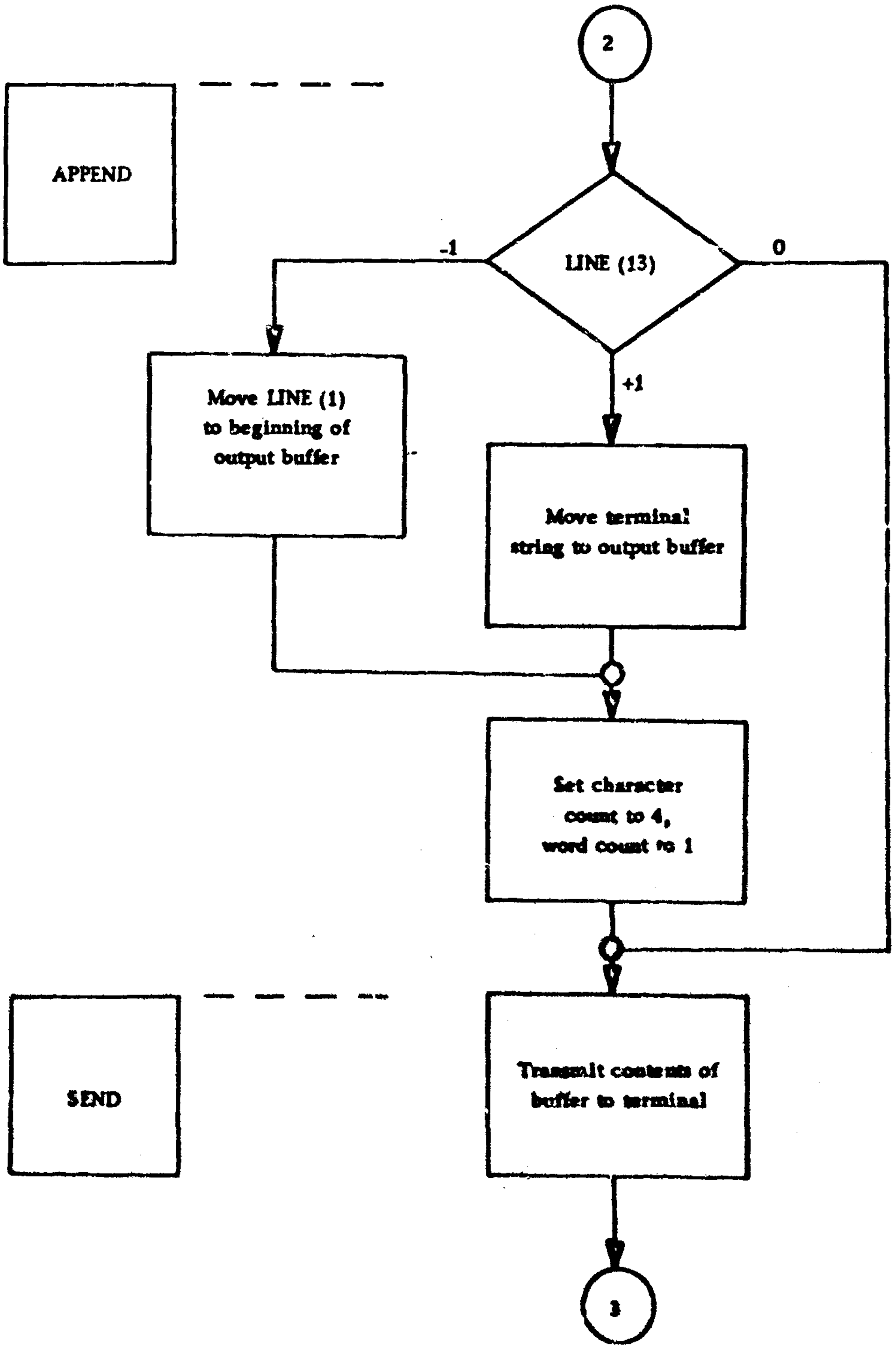

2
APPEND
LINE (13)
-1
0
+1
Move LINE (1) to beginning of output buffer
Move terminal string to output buffer
Set character count to 4, word count to 1
SEND
Transmit contents of buffer to terminal
3

Figure VII-4. BELCH (Cont'd.)

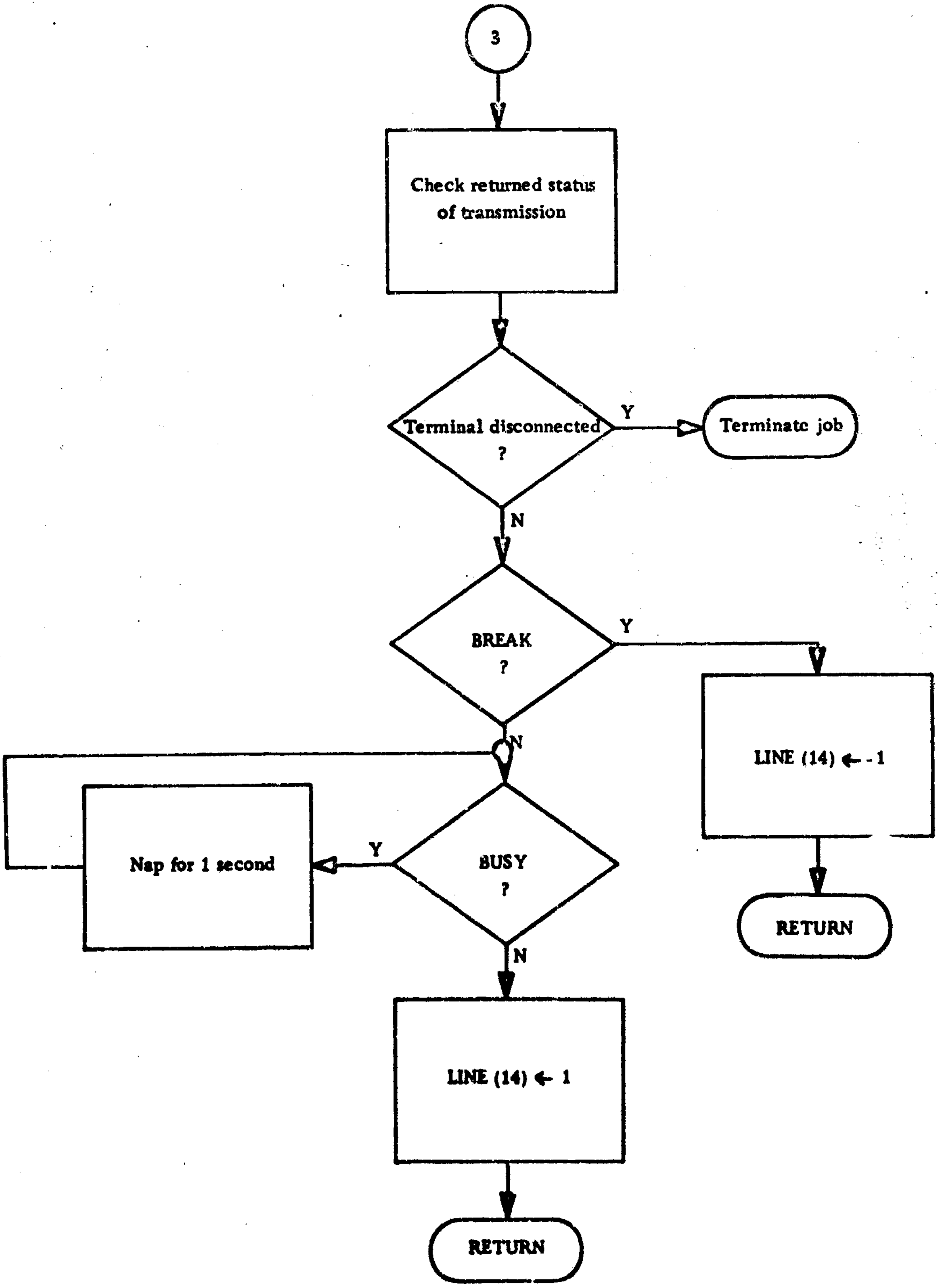


Figure VII-4. BELCH (Concluded)

the on-line system. If the terminal operator has sent a BREAK, then LINE (14) is set to -1 and control is returned to DIALOGUE; if the operation terminates normally, LINE (14) is set to +1 and control is returned to DIALOGUE.

VII. 5. 2 GULP

Subprogram GULP also uses array LINE of labelled common area CUD to communicate with DIALOGUE, in this manner:

Position	Contents
1-12	Data received from terminal.
13	Termination status.

The data read from the terminal are presented to DIALOGUE in the GE 6-bit character set, six characters to a word.

The line is left-justified within LINE; unused space to the right of the message is filled with blanks. LINE (13) is set to 1 if the read operation was concluded successfully; if the operator has sent a BREAK, then LINE (13) is set to -1. Two conditions can cause GULP to terminate this run of the on-line system without returning control to DIALOGUE: disconnection of the terminal during the read operation, or disconnection of the terminal before GULP is called.

Figure VII-5 is a flowchart of GULP. The squares to the left of the flowchart give the names of the program segments.

VII. 5. 3 SELECT

The operation of subprogram OUT is flowcharted in Figure VII-6. OUT is called by DIALOGUE with a single parameter, MESS, which specifies which standard system message is to be transmitted to the remote terminal. OUT then places the specified message in array LINE, and calls BFLCH, returning in MESS the status indicator that BELCH placed in LINE (14)

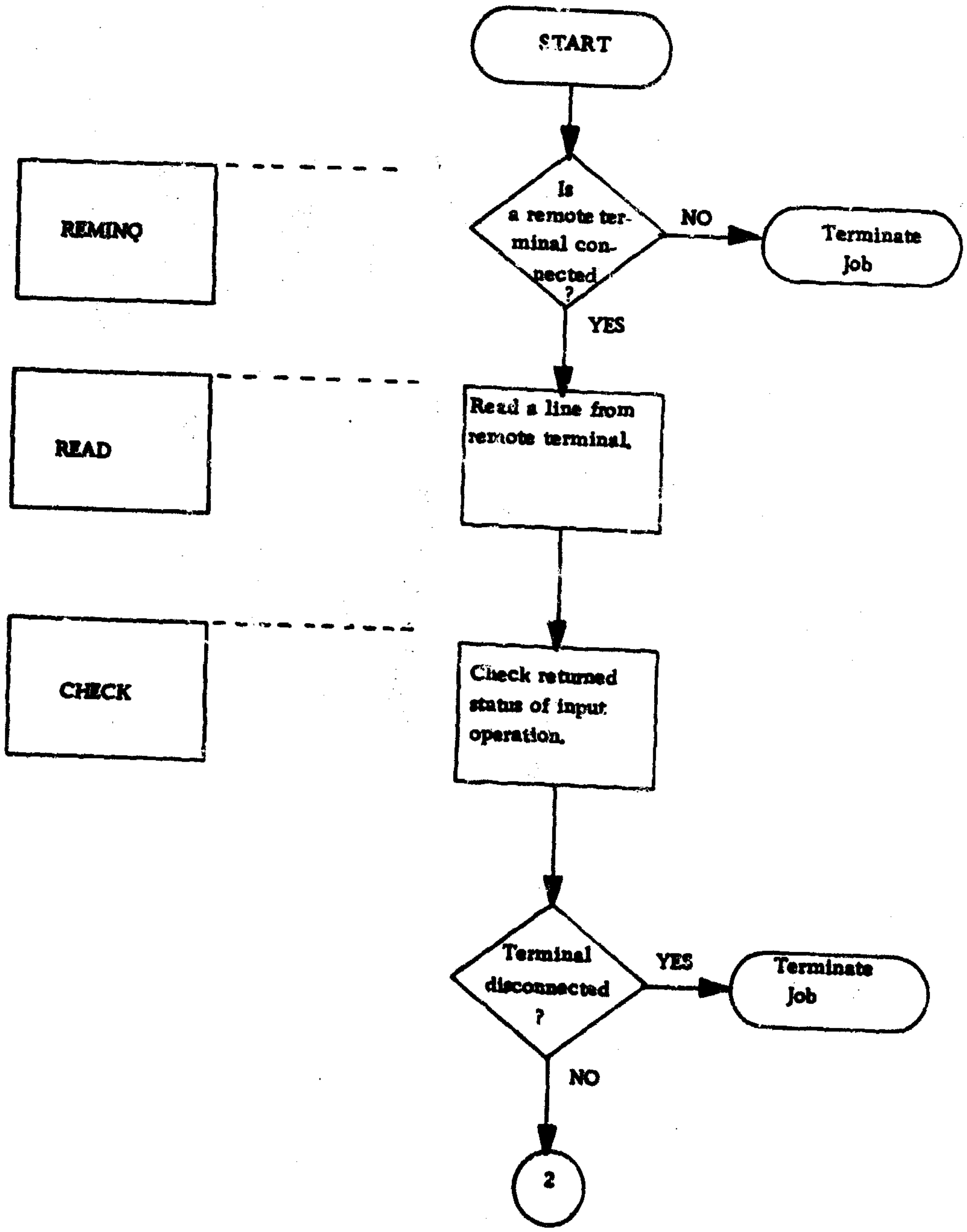

START
REMINQ
Is a remote terminal connected ?
NO
Terminate Job
YES
READ
Read a line from remote terminal.
CHECK
Check returned status of input operation.
Terminal disconnected ?
YES
Terminate Job
NO
2

Figure VII-5. GULP (Cont'd.)

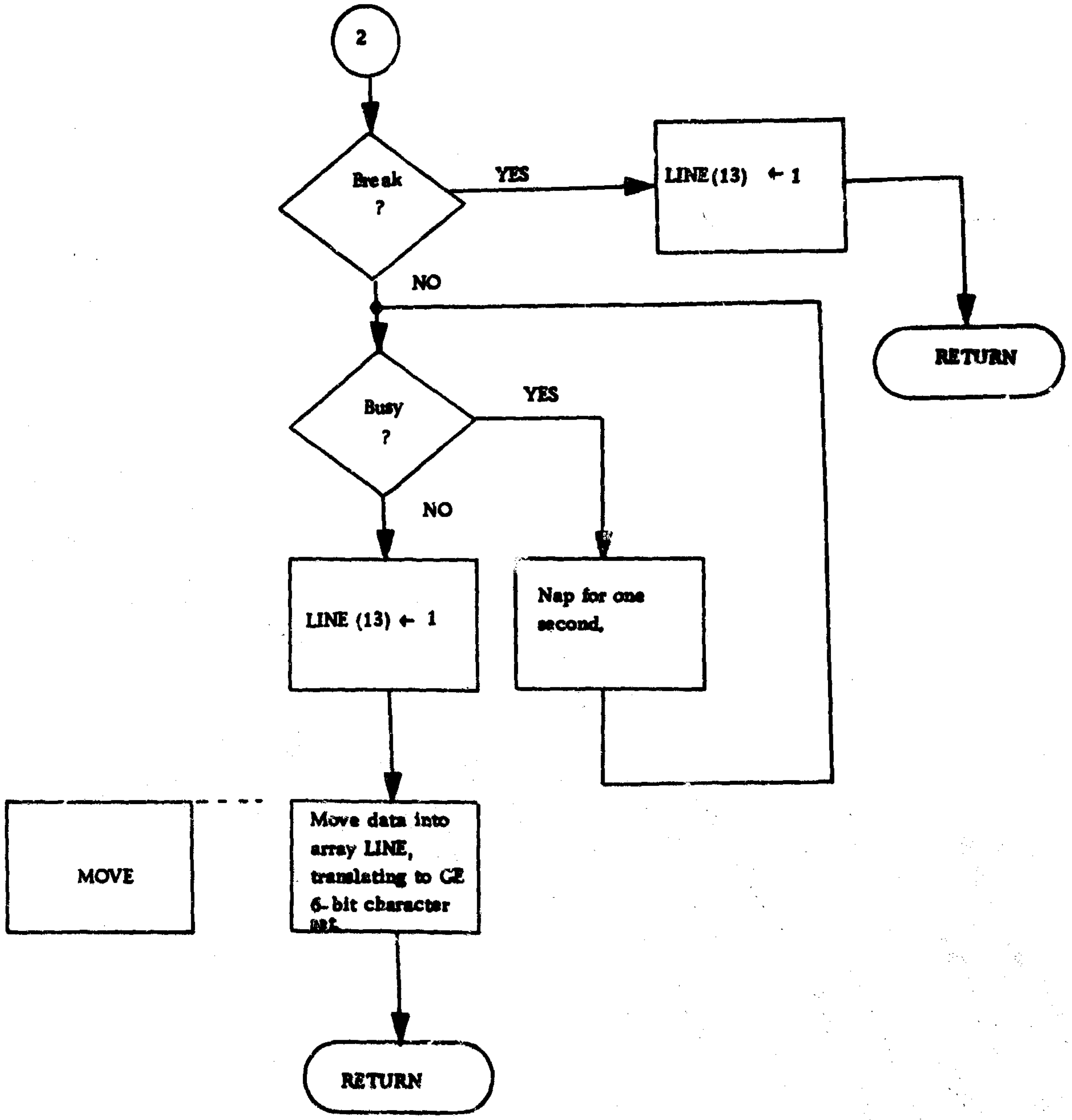


Figure VII-5. GULP (Concluded)

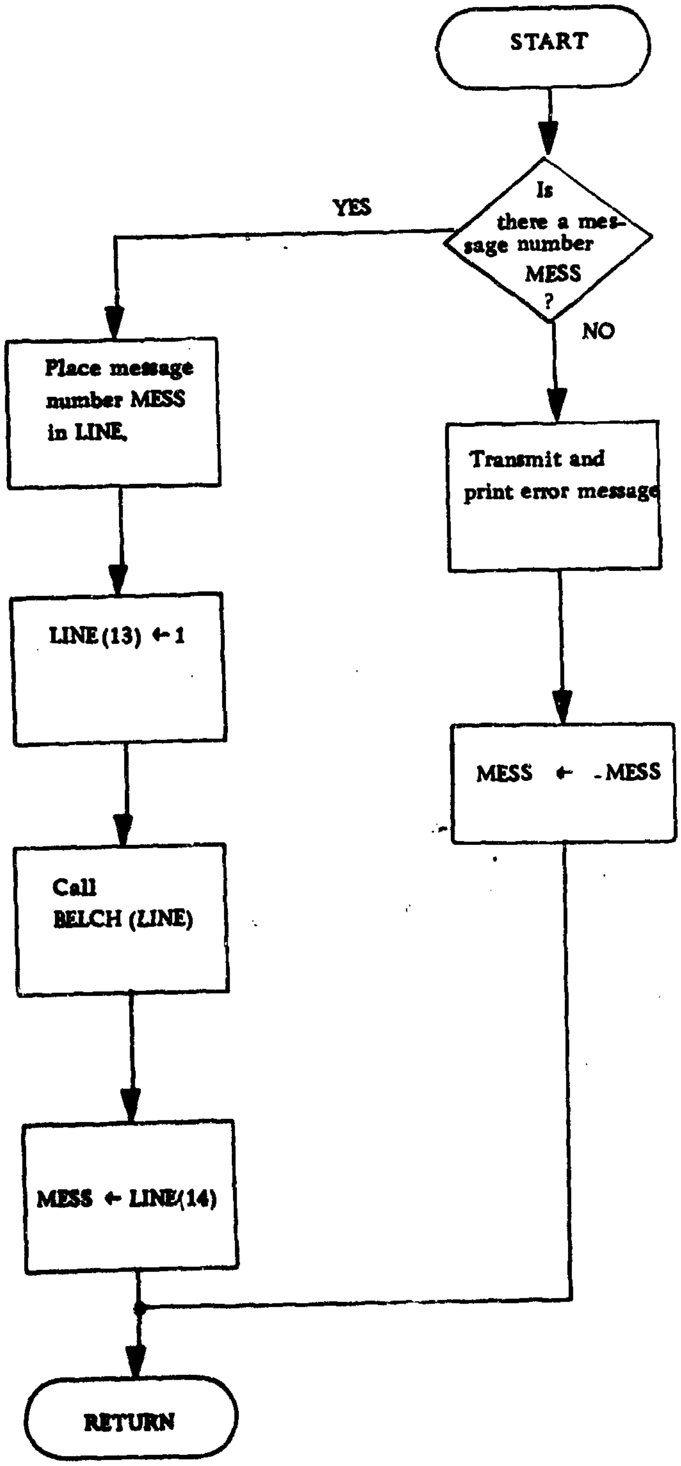

START
Is there a message number MESS ?
YES
NO
Place message number MESS in LINE.
LINE(13) ← 1
Call BELCH (LINE)
MESS ← LINE(14)
Transmit and print error message
MESS ← -MESS
RETURN

Figure VII-6. OUT (MESS)

SECTION VIII

REFERENCES

1. Baker, A.W., Hofmann, J. P. and Smith, J. L. Colex User's Manual. System Development Corporation, Technical Memorandum TM-DA-(L)-15/001/00, undated.

2. Becker, Joseph and Hayes, R. M., Information Storage and Retrieval: Tools, Elements, Theories. John Wiley & Sons, New York, London, Sydney; 1967.

3. Berul, Lawrence, Information Storage and Retrieval: A State-of-the-Art Report. Auerbach Corporation, Philadelphia, Pennsylvania; 14 September 1964.

4. Bleier, Robert E., Treating hierarchical data structures in the SDC time-shared data management system (TDMS); Proceedings of 22nd National Conference Association for Computing Machinery, Thompson Book Company, Washington, D. C.; 1967; p. 41.

5. Borko, Harold and Bernick, M.D., Automatic document classification: Part II additional experiments. J. ACM 11 (April 1964), pp. 139-151.

6. Borko, H., Blankenship, D.A. and Burket, Robert C., On-Line information retrieval using associative indexing, RADC TR-68-100, Rome Air Development Center, May 1968, pp. 119-121. AD 670 195

7. Brillouin, Leon, Science and Information Theory. Academic Press, New York, 1956.

8. Brooks, Frederic P., Jr. and Iverson, Kenneth E., Automatic Data Processing. John Wiley, New York, 1963.

9. Cautin, Harvey; Lowe, Thomas C.; Rapp, Fredericka; and Rubinoff, Morris, An Experimental On-Line Information Retrieval System. University of Pennsylvania, The Moore School of Electrical Engineering, Philadelphia, Pennsylvania, May 1967.

10. Cherry, Colin, On Human Communication. The MIT Press, Massachusetts Institute of Technology, Cambridge, Mass., 1966.

11. Cuadra, Carlos A., ed., Annual Review of Information Science and Technology, V. 2. Interscience Publishers, a division of John Wiley & Sons, New York, 1967.

12. Cuadra, Carlos A., ed., Annual Review of Information Science and Technology, V. 3. Encyclopedia Britannica, Inc., Chicago, 1968.

13. Dattola, R. T., A fast algorithm for automatic classification, in Rep. No. ISR-14 to the National Science Foundation, Cornell University, Ithaca, New York (June 1967).

14. Dennis, Sally F., The design and testing of a fully automatic indexing-searching system for documents consisting of expository text, Information Retrieval A Critical View, G. Schecter, ed., Thompson Book Company, 1967, p. 67.

15. Doyle, L. B., Breaking the cost barrier in automatic classification, SDC Paper DP-2515, July 1966, pp. 29-30.

16. Doyle, L. B., Computing Review 16,841. Review Journal of the ACM 10, 6 (June 1969) pp. 271-272.

17. Feller, William, An Introduction to Probability Theory and its Application, V. 1. Wiley, New York, 1957.

18. Giering, Richard H., Information Processing and the Data Spectrum, Data Corporation publication (October 1967).

19. Giering, Richard H. , Analysis of Existing and Proposed Data Handling Systems. Data Corp. Technical Note DTN-69-9 (October 1967).

20. Haibt, Luther, Fischer, Margaret, Kastner, Margaret, Ketelhut, Robert, Ogg, Jay, and Woolley, Jon H., Retrieving 4000 references without indexing. The Fourth Annual National Colloquium on Information Retrieval, A. B. Tonik, ed., Internal Information Incorporated, Philadelphia, Pennsylvania, 1967; p. 127.

21. Hays, David C., Introduction to Computational Linguistics, American Elsevier, New York, 1968, pp. 85-105.

22. Head, Robert V., Dribble posting a master file. C. ACM 9, 2 (February 1966) pp. 106-107.

23. Ide, E., Williamson, R. and Williamson, D., The Cornell programs for cluster searching and relevance feedback, in Rep. No. ISR-12 to the National Science Foundation, Cornell University, Ithaca, New York, June 1967, pp. IV-1--IV-13.

24. Jones, P. E., et. al., Papers on Automatic Language Processing: Selected Collection Data Analyses. Volume 1. Arthur D. Little, Inc. Cambridge, Mass. (February, 1967), AD 649 073.

25. Jones, K.S and Jackson, D., Current approaches to classification and clump-finding at the Cambridge Language Research Unit. The Computer Journal 10, (May 1967), pp. 29-37.

26. Landau, Robert M., Convergence to higher relevance retrieval by the use of on-line query techniques. Proceedings of the American Society for Information Sciences, Greenwood Publishing Corporation, New York, p. 189 (1963).

27. Lesser, V. R., A modified search algorithm using requesting clustering in Rep. No. ISR-11 to the National Science Foundation, Cornell University, Ithaca, New York, June 1966, p. VII-1.

28. Lowe, Thomas C., Direct-access memory retrieval using truncated record names, Software Age 1, 1(September 1967), 28.

29. Lowe, Thomas C., Encoding from alphanumeric names to record addresses, Software Age 2, 3 (April 1968), 15.

30. Lowe, Thomas C., The influence of data base characteristics and usage in direct-access file organization, J. ACM 15, 4 (Oct. 1968) p. 535-548.

31. Lowe, Thomas C., VanDyke, J. Gary O. and Colilla, Robert A., Program Paging and Operating Algorithms. Rome Air Development Center Technical Report No. RADC-TR-68-444 (1968); AD-845-758.

32. Lowe, Thomas C., Automatic segmentation of cyclic program structures based on connectivity and processor timing. RADC approval SCES-69-907; accepted for publication in C. ACM.

33. Luhn, H. P, The automatic creation of literature abstracts. IBM J. (April 1958) pp. 159-165.

34. Maron, M.E., Automatic indexing: an experimental inquiry. J. ACM 8 (July 1961), pp. 404-417.

35. Meadow, Charles T., The Analysis of Information Systems. John Wiley & Sons, New York, 1967.

36. Nance, J. W. and Lathrop, J. W., System Design Specification General Purpose Orbit. System Development Corporation Technical Memorandum TM-DA-20/000/00 (1968).

37. National Science Foundation, Nonconventional Scientific and Technical Information Systems in Current Use. Report No. NSF-66-24 (December 1966).

38. Rocchio, Joseph John Jr., Document Retrieval Systems-Optimization and Evaluation. Ph.D. Dissertation, Harvard University (1966). (Also published as Harvard Computation Laboratory Report to the National Science Foundation No. ISR-10).

39. Rubinoff, Morris, et. al., Summary Description of Easy English. University of Pennsylvania, The Moore School of Electrical Engineering, February, 1967.

40. Salton, G. and Lesk, M.E., The SMART automatic document retrieval system - An illustration. C.ACM 8, 6(June 1965), pp. 391-398.

41. Salton, Gerard, The evaluation of automatic retrieval procedures - selected test results using the SMART system. American Documentation 16, 3(July 1965) pp. 209-222.

42. Salton, G., et. al., The SMART system - retrieval results and future plans, in Rep. No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, Ithaca, New York, June 1966, I-4.

43. Salton, G. and Lesk, M.E., Computer evaluation of indexing and text processing. J.ACM 15, 1 (Jan. 1967) 8-36.

44. Salton, G., Automatic Information Organization and Retrieval. McGraw-Hill, New York, 1968.

45. Sammon, John W., Jr., Some Mathematics of Information Storage and Retrieval. Rome Air Development Center Report RADC-TR-68-178 (June 1968). AD 673 362

46. Shannon, C.E., Prediction and entropy in printed English. Bell System Technical J. 30 (1951).

47. Stevens, Mary Elizabeth, Automatic Indexing: A State of the Art Report. National Bureau of Standards Monograph 91 (1965).

48. Stiles, H. Edmund, Automatic indexing and the association factor, in Information Systems Compatibility, Simon M. Newman, Ed., Sparton Books, Washington, D. C., 1964.

49. Stone, Don C., Word statistics in the generation of semantic tools for information systems, University of Pennsylvania, The Moore School of Electrical Engineering, Report No. 68-32 (December 1967); AD-664-915.

50. Stone, Don. C. and Rubinoff, Morris, Statistical generation of a technical vocabulary (brief communication). American Documentation 19 (October 1968).

51. Verhoeff J., Goffman, W. and Belzer, Jack, Inefficiency of the use of Boolean functions for information retrieval systems. C.ACM 4, 12 (December 1961) p. 557-559.

52. Welch, Noreen O., A Survey of Five On-Line Retrieval Systems. Mitre Corporation Report MTP-322 (August 1968) in support of COSATI Panel 2.

53. Zipf, George Kingsley, Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, Mass., 1949.

54. Zunde, Pranas and Dexter, Margaret E., Statistical models of index vocabularies. Proceedings of the American Society for Information Science. 5 (October 1968) 73.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Informatics Inc.
4720 Montgomery Lane
Bethesda, Maryland 20014

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
ON-LINE RETRIEVAL

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Interim Report

5. AUTHOR(S) (First name, middle initial, last name)
Thomas C. Lowe
David C. Roberts

6. REPORT DATE November 1969	7a. TOTAL NO. OF PAGES 212	7b. NO. OF REFS 54

8a. CONTRACT OR GRANT NO.
F30602-69-C-0038
b. PROJECT NO.
4594
c. Task No.
459401
d.

9a. ORIGINATOR'S REPORT NUMBER(S)
TR-69-1090-1

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
RADC-TR-69-304

10. DISTRIBUTION STATEMENT
This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made only with prior approval of RADC (EMIDB), GAFB, N.Y. 13440.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Rome Air Development Center (EMIDB)
Griffiss Air Force Base, New York 13440

13. ABSTRACT

The report is concerned with the development of an on-line information storage and retrieval system for the Rome Air Development Center. It is deeply concerned with automatic document classification [illegible] ractive retrieval and the problems peculiar to the automated processin[illegible] rge document collections. In the light of the requirements to be met, a report on existing applicable techniques is made. This is followed by a description of other methods, developed specifically for this project. Finally, the system design is presented to a detailed level.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Information storage and retrieval						
On-line retrieval						
Interactive retrieval						
Man-machine dialogue						
Automatic thesaurus generation						
Concept identification						
Concept vectors						
Concept clustering						
Stem analysis						
Common words						
Document clustering						
Concordances						
Project SMART						
Automatic indexing						
Correlation measures						

UNCLASSIFIED
Security Classification